A brief information about the dataset

McDonald's Nutrition Facts is a dataset from Kaggle
McDonald's food items is always controversial because of their high level of sodium content in them.
The aim of this project is to perform nutritional analysis of the items on the menu.
And also to categorize items according to factors like sugar and fibre content. Then modelling the results using the
bar and pie charts, scatter plots, boxplots and heatmap plots.

In [2]:
```
# Importing the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
Data = pd.read_csv('menu.csv') #reading out the csv file
```

In [5]:
```
pd.set_option('display.max_columns', None)
```

In [6]:
```
# Displaying the first top 5 rows of the dataset
Data.head()
```

Out[6]:

	Category	Item	Serving Size	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	Choles
0	Breakfast	Egg McMuffin	4.8 oz (136 g)	300	120	13.0	20	5.0	25	0.0	
1	Breakfast	Egg White Delight	4.8 oz (135 g)	250	70	8.0	12	3.0	15	0.0	
2	Breakfast	Sausage McMuffin	3.9 oz (111 g)	370	200	23.0	35	8.0	42	0.0	
3	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	450	250	28.0	43	10.0	52	0.0	
4	Breakfast	Sausage McMuffin with Egg Whites	5.7 oz (161 g)	400	210	23.0	35	8.0	42	0.0	

In [7]:
```
#To check the number of rows and columns contained in the dataset
Data.shape
```

Out[7]:
```
(260, 24)
```

There are 260 rows and 24 columns contained in the dataset

In [8]:
```
# Getting the information about the dataset
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 260 entries, 0 to 259
Data columns (total 24 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Category                        260 non-null    object 
 1   Item                            260 non-null    object 
 2   Serving Size                    260 non-null    object 
 3   Calories                        260 non-null    int64  
 4   Calories from Fat               260 non-null    int64  
 5   Total Fat                       260 non-null    float64
 6   Total Fat (% Daily Value)       260 non-null    int64  
 7   Saturated Fat                   260 non-null    float64
 8   Saturated Fat (% Daily Value)   260 non-null    int64  
 9   Trans Fat                       260 non-null    float64
 10  Cholesterol                     260 non-null    int64  
 11  Cholesterol (% Daily Value)     260 non-null    int64  
 12  Sodium                          260 non-null    int64  
 13  Sodium (% Daily Value)          260 non-null    int64  
 14  Carbohydrates                   260 non-null    int64  
 15  Carbohydrates (% Daily Value)   260 non-null    int64  
 16  Dietary Fiber                   260 non-null    int64  
 17  Dietary Fiber (% Daily Value)   260 non-null    int64  
 18  Sugars                          260 non-null    int64  
 19  Protein                         260 non-null    int64  
 20  Vitamin A (% Daily Value)       260 non-null    int64  
 21  Vitamin C (% Daily Value)       260 non-null    int64  
 22  Calcium (% Daily Value)         260 non-null    int64  
 23  Iron (% Daily Value)            260 non-null    int64  
dtypes: float64(3), int64(18), object(3)
memory usage: 48.9+ KB
```

Observations

- The datasets were all place in their respective datatype

In [9]:
```
# checking for null columns in the dataset
Data.isnull().sum()
```

Out[9]:
```
Category                          0
Item                              0
Serving Size                      0
Calories                          0
Calories from Fat                 0
Total Fat                         0
Total Fat (% Daily Value)         0
Saturated Fat                     0
Saturated Fat (% Daily Value)     0
Trans Fat                         0
Cholesterol                       0
Cholesterol (% Daily Value)       0
Sodium                            0
Sodium (% Daily Value)            0
Carbohydrates                     0
Carbohydrates (% Daily Value)     0
Dietary Fiber                     0
Dietary Fiber (% Daily Value)     0
Sugars                            0
Protein                           0
Vitamin A (% Daily Value)         0
Vitamin C (% Daily Value)         0
Calcium (% Daily Value)           0
Iron (% Daily Value)              0
dtype: int64
```

Observations

- There is no null values in the dataset for all the columns

In [10]:
```
# Checking for the consistency of the columns names
Data.columns
```

Out[10]:
```
Index(['Category', 'Item', 'Serving Size', 'Calories', 'Calories from Fat',
       'Total Fat', 'Total Fat (% Daily Value)', 'Saturated Fat',
       'Saturated Fat (% Daily Value)', 'Trans Fat', 'Cholesterol',
       'Cholesterol (% Daily Value)', 'Sodium', 'Sodium (% Daily Value)',
       'Carbohydrates', 'Carbohydrates (% Daily Value)', 'Dietary Fiber',
       'Dietary Fiber (% Daily Value)', 'Sugars', 'Protein',
       'Vitamin A (% Daily Value)', 'Vitamin C (% Daily Value)',
       'Calcium (% Daily Value)', 'Iron (% Daily Value)'],
      dtype='object')
```

In [11]:
```
# Checking for the uniqueness of the dataset
Data.nunique()
```

Out[11]:
```
Category                          9
Item                            260
Serving Size                    107
Calories                         78
Calories from Fat                48
Total Fat                        52
Total Fat (% Daily Value)        63
Saturated Fat                    26
Saturated Fat (% Daily Value)    74
Trans Fat                         5
Cholesterol                      35
Cholesterol (% Daily Value)      48
Sodium                          111
Sodium (% Daily Value)           65
Carbohydrates                    84
Carbohydrates (% Daily Value)    40
Dietary Fiber                     8
Dietary Fiber (% Daily Value)    24
Sugars                           83
Protein                          42
Vitamin A (% Daily Value)        19
Vitamin C (% Daily Value)        17
Calcium (% Daily Value)          16
Iron (% Daily Value)             12
dtype: int64
```

In [13]:
```
Data['Serving Size'].unique()
```

Out[13]:
```
array(['4.8 oz (136 g)', '4.8 oz (135 g)', '3.9 oz (111 g)',
       '5.7 oz (161 g)', '6.5 oz (185 g)', '5.3 oz (150 g)',
       '5.8 oz (164 g)', '5.4 oz (153 g)', '5.9 oz (167 g)',
       '4.1 oz (117 g)', '4.6 oz (131 g)', '5.7 oz (163 g)',
       '6.2 oz (177 g)', '6.4 oz (181 g)', '5 oz (143 g)',
       '5.5 oz (157 g)', '7.1 oz (201 g)', '6.1 oz (174 g)',
       '6.3 oz (178 g)', '5 oz (141 g)', '7.2 oz (205 g)',
       '6.9 oz (197 g)', '8.5 oz (241 g)', '9.5 oz (269 g)',
       '10 oz (283 g)', '9.6 oz (272 g)', '10.1 oz (286 g)',
       '14.8 oz (420 g)', '15.3 oz (434 g)', '14.9 oz (423 g)',
       '15.4 oz (437 g)', '5.3 oz (151 g)', '6.8 oz (192 g)',
       '2 oz (56 g)', '4 oz (114 g)', '9.6 oz (251 g)', '7.4 oz (211 g)',
       '7.1 oz (202 g)', '8 oz (227 g)', '8.3 oz (235 g)',
       '8.6 oz (244 g)', '3.5 oz (98 g)', '4 oz (113 g)',
       '9.5 oz (270 g)', '5.2 oz (147 g)', '6.7 oz (190 g)',
       '5.6 oz (159 g)', '7.3 oz (208 g)', '7.5 oz (213 g)',
       '7 oz (200 g)', '8.8 oz (249 g)', '8.1 oz (230 g)',
       '7.6 oz (217 g)', '10 oz (284 g)', '5.6 oz (160 g)',
       '5.1 oz (143 g)', '6 oz (171 g)', '5.2 oz (148 g)',
       '11.1 oz (316 g)', '10.7 oz (302 g)', '10.9 oz (310 g)',
       '10.5 oz (297 g)', '11.1 oz (314 g)', '11.2 oz (318 g)',
       '10.7 oz (304 g)', '10.3 oz (291 g)', '2.3 oz (65 g)',
       '3.4 oz (97 g)', '5.7 oz (162 g)', '11.4 oz (323 g)',
       '22.8 oz (646 g)', '5 oz (142 g)', '7.9 oz (223 g)',
       '9 oz (255 g)', '12.3 oz (348 g)', '11.8 oz (335 g)',
       '4.6 oz (130 g)', '4.3 oz (123 g)', '4.1 oz (116 g)',
       '4.5 oz (128 g)', '4.3 oz (121 g)', '2.6 oz (75 g)',
       '5.9 oz (168 g)', '1.3 oz (38 g)', '3.1 oz (87 g)',
       '1.2 oz (34 g)', '5.2 oz (149 g)', '2.7 oz (77 g)',
       '1 cookie (33 g)', '1 oz (29 g)', '6.3 oz (179 g)',
       '6.4 oz (182 g)', '16 fl oz cup', '21 fl oz cup', '30 fl oz cup',
       '12 fl oz cup', '1 carton (236 ml)', '6 fl oz (177 ml)',
       '22 fl oz cup', '16.9 fl oz', '20 fl oz cup', '32 fl oz cup',
       '16.2 oz (460 g)', '7.3 oz (207 g)', '10.1 oz (285 g)',
       '13.4 oz (381 g)', '14.2 oz (403 g)'], dtype=object)
```

In [25]:
```python
# working on the serving size column to convert everything to grams (g)

import re

# Define the regular expression pattern for extracting values in brackets
pattern = re.compile(r'\((.*?)\)')

# Apply the pattern to the 'Serving size' column and extract the values
Data['Serving Size'] = Data['Serving Size'].apply(lambda x: pattern.findall(x)[0] if p

# function to extract and convert values
def serve(size):
    if ' g' in size:
        return float(size.replace(' g','')) # return value in g
    elif 'fl oz' in size:
        x, y = size.split(' fl oz')
        return round(float(float(x) * 29.574)) # 1 fl oz == 29.574 ml == 29.574g
    elif ' ml' in size:
        return float(size.replace(' ml','')) # 1ml == 1g
```

In [26]:
```python
# apply and rename column
Data['Serving Size'] = Data['Serving Size'].apply(serve)
Data = Data.rename(columns = {'Serving Size': 'Serving Size (g)'})
```

In [27]:
```
# check for changes made

Data['Serving Size (g)'].unique()
```

Out[27]:
```
array([136., 135., 111., 161., 185., 150., 164., 153., 167., 117., 131.,
       163., 177., 181., 143., 157., 201., 174., 178., 141., 205., 197.,
       241., 269., 283., 272., 286., 420., 434., 423., 437., 151., 192.,
        56., 114., 251., 211., 202., 227., 235., 244.,  98., 113., 270.,
       147., 190., 159., 208., 213., 200., 249., 230., 217., 284., 160.,
       171., 148., 316., 302., 310., 297., 314., 318., 304., 291.,  65.,
        97., 162., 323., 646., 142., 223., 255., 348., 335., 130., 123.,
       116., 128., 121.,  75., 168.,  38.,  87.,  34., 149.,  77.,  33.,
        29., 179., 182., 473., 621., 887., 355., 236., 651., 500., 591.,
       946., 460., 207., 285., 381., 403.])
```

In [ ]:
```
# getting the groups and counts of categories of the dataset
df = Data['Category'].value_counts()
df
```

In [ ]:
```
# visualizing the value counts of category
df = Data['Category'].value_counts().reset_index()
df
# Creating a bar chart
sns.barplot(x=df['Category'], y=df['index'], color='grey')
plt.xlabel('Items')
plt.ylabel('Category')
plt.title('Count of items in each category')
```

Observations

- There are 9 Unique categories on the category column
- We have more Items on our menu for Coffee & Tea, followed by Breakfast.
- Smoothies & Shakes, Chicken & fish and also beverages have approxiamtely equal amount of items.
- Salads has the least items on our menu.

checking for duplicates in all the rows

In [13]:
```
Data.duplicated().sum()
```

Out[13]:
```
0
```

Observation

- There is no duplicate on the dataset

In [14]:
```
Data['Item'].value_counts().sum()
```

Out[14]:
```
260
```

In [15]:
```
Data['Serving Size'].nunique()
```

Out[15]:
```
107
```

In [16]:
```
Data['Serving Size'].value_counts().sum()
```

Out[16]:
```
260
```

- item and serving size column have a total of 260 == nrow size

Exploratory Data Analysis (EDA)

In [17]:
```
# Checking the statistical summarry of the numerical variables in the dataset
Data.describe()
```

Out[17]:

	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	Choleste
count	260.000000	260.000000	260.000000	260.000000	260.000000	260.000000	260.000000	260.000
mean	368.269231	127.096154	14.165385	21.815385	6.007692	29.965385	0.203846	54.942
std	240.269886	127.875914	14.205998	21.885199	5.321873	26.639209	0.429133	87.269
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	210.000000	20.000000	2.375000	3.750000	1.000000	4.750000	0.000000	5.000
50%	340.000000	100.000000	11.000000	17.000000	5.000000	24.000000	0.000000	35.000
75%	500.000000	200.000000	22.250000	35.000000	10.000000	48.000000	0.000000	65.000
max	1880.000000	1060.000000	118.000000	182.000000	20.000000	102.000000	2.500000	575.000

8 rows × 21 columns

- From the statistical summary, sodium has a maximum sodium content of 3600.
- And an average sodium content of 495.75 on the menu

Separating the numerical and the categorical variables for easy analysis

In [18]:
```
Data.select_dtypes(include=object).columns.tolist()
```

Out[18]:
```
['Category', 'Item', 'Serving Size']
```

In [19]:
```
Data.select_dtypes(include=np.number).columns.tolist()
```

Out[19]:
```
['Calories',
 'Calories from Fat',
 'Total Fat',
 'Total Fat (% Daily Value)',
 'Saturated Fat',
 'Saturated Fat (% Daily Value)',
 'Trans Fat',
 'Cholesterol',
 'Cholesterol (% Daily Value)',
 'Sodium',
 'Sodium (% Daily Value)',
 'Carbohydrates',
 'Carbohydrates (% Daily Value)',
 'Dietary Fiber',
 'Dietary Fiber (% Daily Value)',
 'Sugars',
 'Protein',
 'Vitamin A (% Daily Value)',
 'Vitamin C (% Daily Value)',
 'Calcium (% Daily Value)',
 'Iron (% Daily Value)']
```

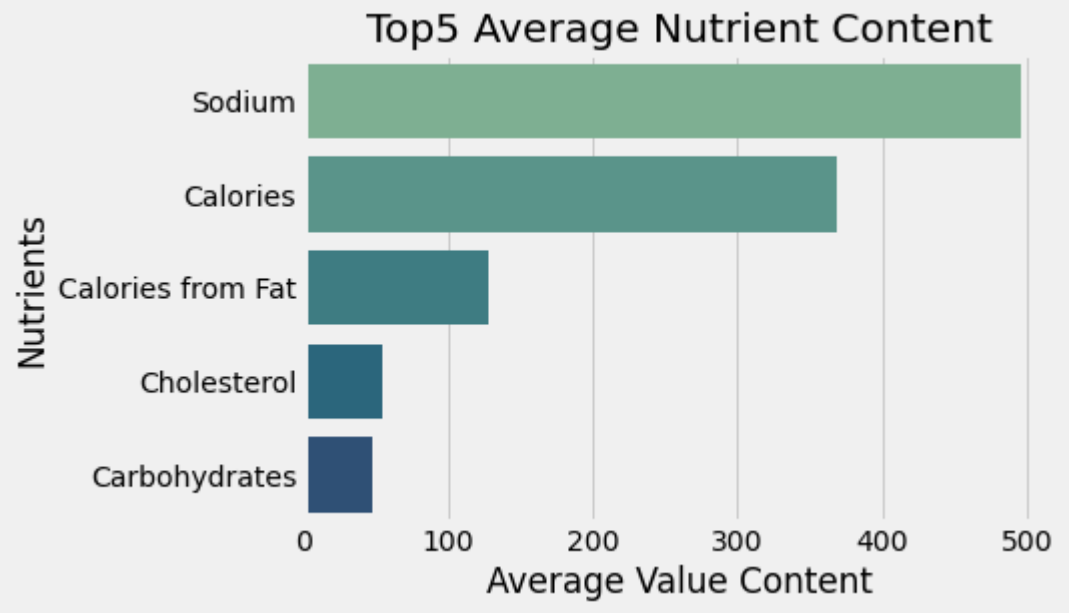
Checking for the top5 nutrient content on the menu

In [20]:
```
avg_nutrients = Data.mean().sort_values(ascending=False).reset_index()
Top5 = avg_nutrients.head()
```

In [24]:
```
# visualizing the top average nutrient content in the menu
sns.barplot( y = 'index', x=0, data=Top5, palette = 'crest')

plt.ylabel('Nutrients')
plt.xlabel('Average Value Content')
plt.title('Top5 Average Nutrient Content') # Chart label
```

Out[24]:
```
Text(0.5, 1.0, 'Top5 Average Nutrient Content')
```



Observations

- The bar chart shows the top 5 average nutrients.
- The average quantity of sodium we get from our menu is higher than the others,
- i.e the amount of sodium you can get from the items in our menu is high.
- Our top 3 nutrients with High values are Sodium, Calories and Calories from Fat.

Checking the average value of sodium content in all categories on the menu

In [25]:
```
#
Avg_Sodium = Data.groupby('Category')['Sodium'].mean().round().to_frame().sort_values(
Avg_Sodium
```

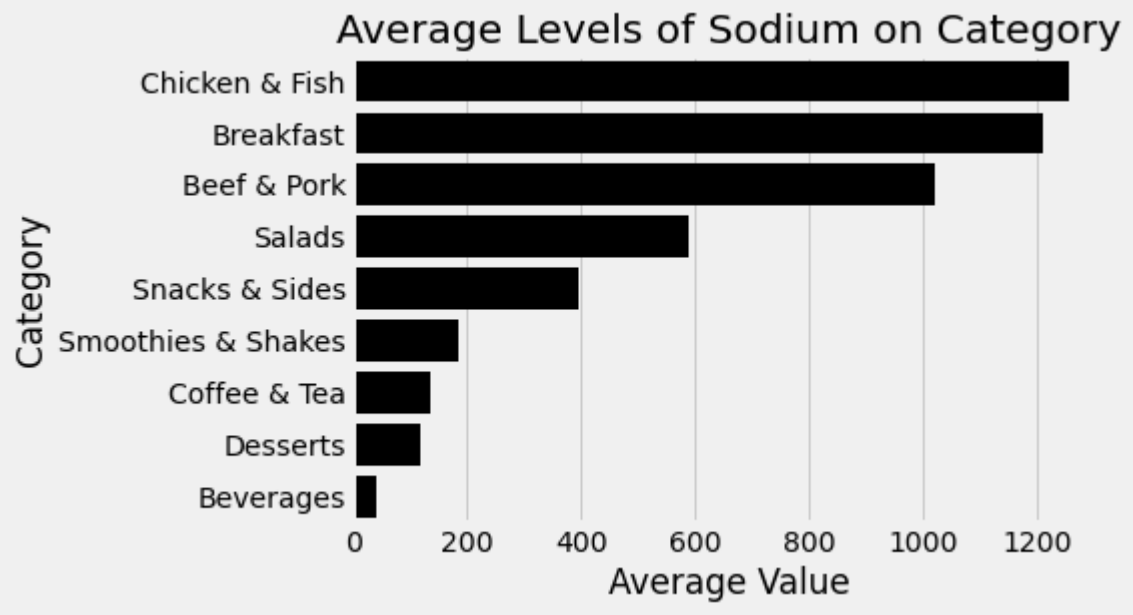
Out[25]:

	Category	Sodium
0	Chicken & Fish	1258.0
1	Breakfast	1211.0
2	Beef & Pork	1021.0
3	Salads	588.0
4	Snacks & Sides	396.0
5	Smoothies & Shakes	184.0
6	Coffee & Tea	137.0
7	Desserts	117.0
8	Beverages	41.0

In [26]:
```
# visualizing the average level of sodium content for category in the menu
sns.barplot(x='Sodium', y = 'Category', data = Avg_Sodium, color='black')

plt.ylabel('Category')
plt.xlabel('Average Value')
plt.title('Average Levels of Sodium on Category') # chart label
```

Out[26]:
```
Text(0.5, 1.0, 'Average Levels of Sodium on Category')
```



Observations

- Chicken & Fish has the highest average level of sodium content with above 1,200
- Breakfast and Beef & Pork at 1,200 and above 1,000 respectively follows Chicken&Fish on average level of sodium content
- Desserts and Beverages have the least average level of sodium content and minimium ways of getting sodium

Checking for the top5 items with the highest level of sodium content on the menu

In [27]:
```
# To get the top5 items with the average amount of sodium content
sod_level = Data.groupby('Item')['Sodium'].max().round().to_frame().sort_values(by= 'S
sod_level
```

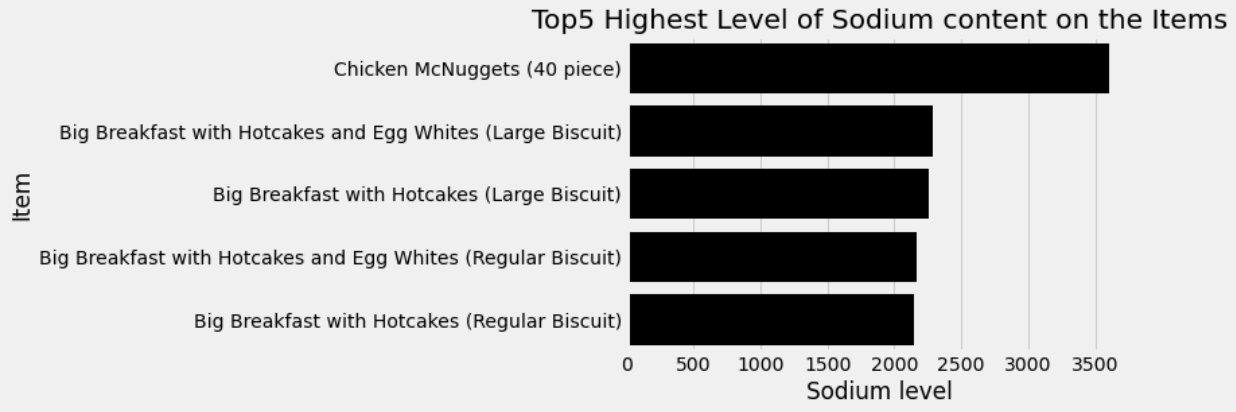
Out[27]:

	Item	Sodium
0	Chicken McNuggets (40 piece)	3600
1	Big Breakfast with Hotcakes and Egg Whites (La...	2290
2	Big Breakfast with Hotcakes (Large Biscuit)	2260
3	Big Breakfast with Hotcakes and Egg Whites (Re...	2170
4	Big Breakfast with Hotcakes (Regular Biscuit)	2150

In [29]:
```
# Visualizing the the top 5 items with the highest average sodium content on the menu
sns.barplot(y='Item', x='Sodium',data= sod_level, color = 'black')

plt.ylabel('Item')
plt.xlabel('Sodium level')
plt.title('Top5 Highest Level of Sodium content on the Items')
```

Out[29]:
```
Text(0.5, 1.0, 'Top5 Highest Level of Sodium content on the Items')
```



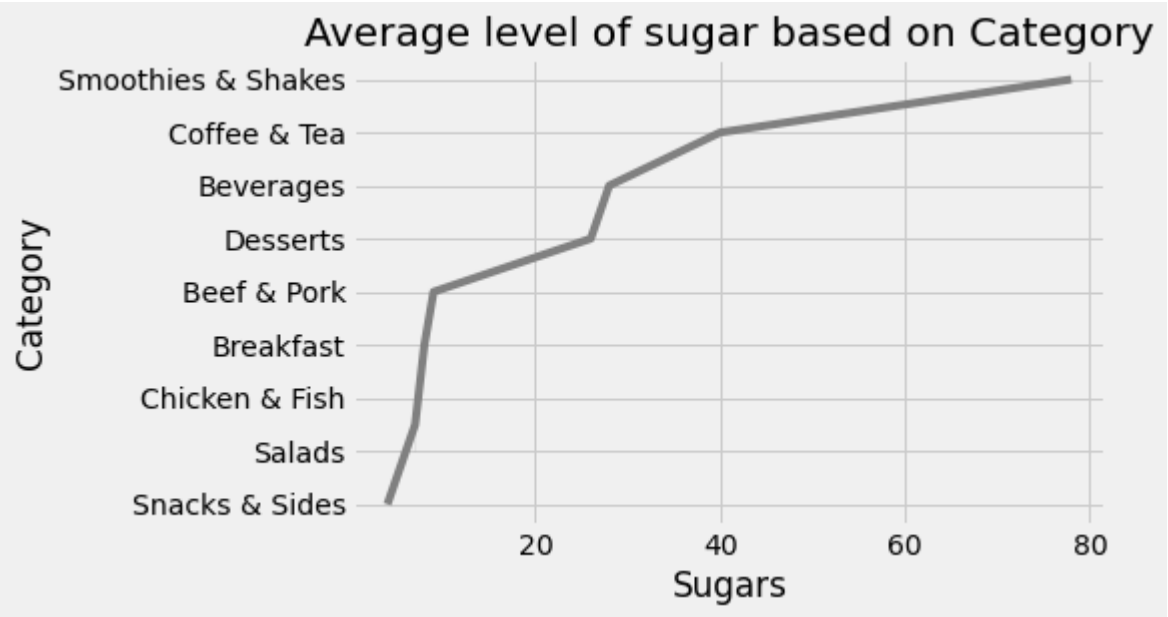
Observations

- Chicken McNuggets(40 piece) is the item with the highest level of sodium content on the menu
- Big Breakfast with Hotcakes and Egg Whites (Large Biscuit) and Big Breakfast with Hotcakes (Large Biscuit) have amost equal level of sodium content

Checking for category with the highest and lowest average amount of sugar content on the menu

In [43]:
```
# Grouping Category by sugar content and getting the average level of sugar content fo
df = Data.groupby('Category')['Sugars'].mean().round().to_frame().sort_values(by='Suga
df
# plotting a lineplot to display the Average level of sugar on each category on the me
sns.lineplot(y= 'Category', x= 'Sugars', data = df, color = 'grey')
# Title label
plt.title('Average level of sugar based on Category')
```

Out[43]:
```
Text(0.5, 1.0, 'Average level of sugar based on Category')
```



Observations

- Smoothies & shakes has the highest ways of getting sugar on the menu
- Snacks & Sides has a fair way of getting sugar on the menu

Checking for the top 5 items with the highest content of sugar

In [31]:
```
Sugar_level = Data.groupby('Item')['Sugars'].max().round().to_frame().sort_values(by='

Sugar_level
```

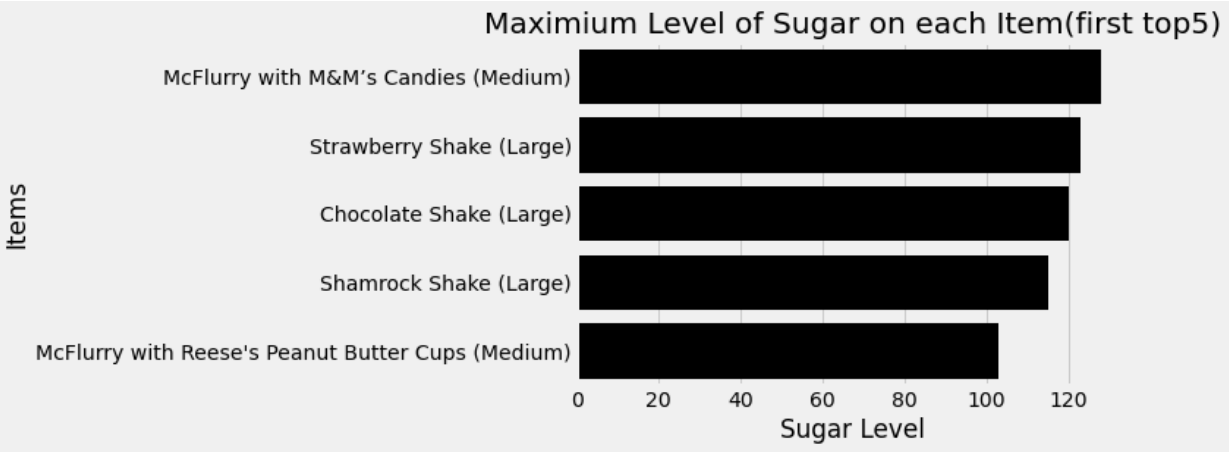
Out[31]:

	Item	Sugars
0	McFlurry with M&M's Candies (Medium)	128
1	Strawberry Shake (Large)	123
2	Chocolate Shake (Large)	120
3	Shamrock Shake (Large)	115
4	McFlurry with Reese's Peanut Butter Cups (Medium)	103

In [32]:
```
# plotting a barplot to show categories of sugar contents on the items
sns.barplot(y='Item', x='Sugars', data = Sugar_level, color = 'black')

plt.xlabel('Sugar Level')
plt.ylabel('Items')
# Chart title
plt.title('Maximium Level of Sugar on each Item(first top5)')
```

Out[32]:
```
Text(0.5, 1.0, 'Maximium Level of Sugar on each Item(first top5)')
```



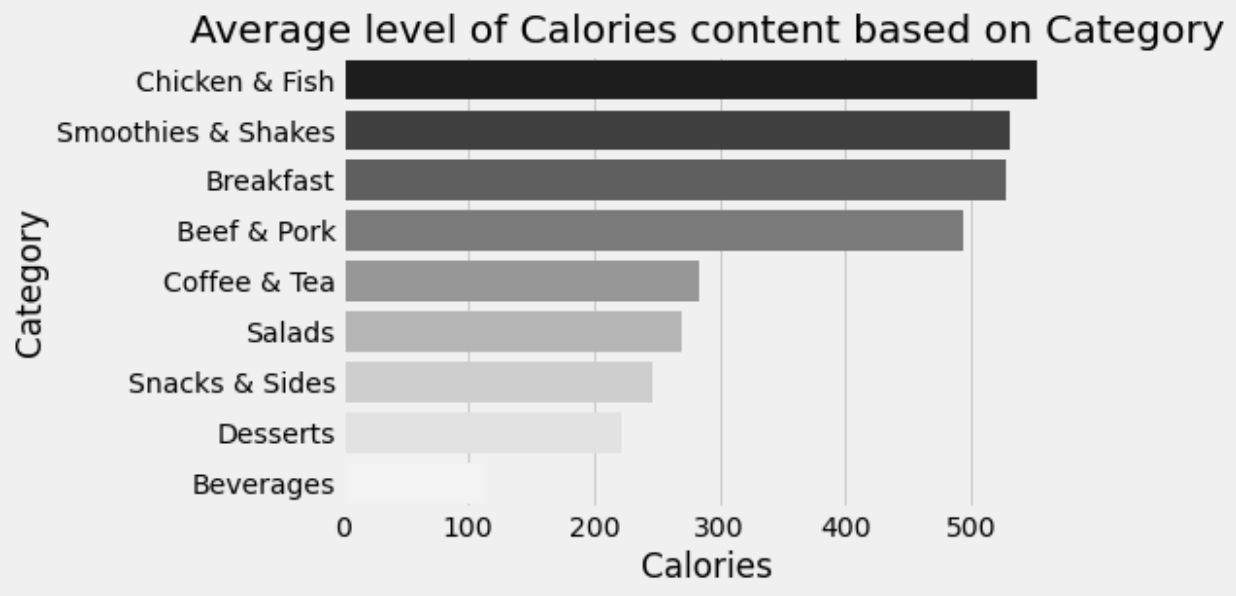
Observation

- McFlurry with M&M's Candies (Medium) is the item that has the highest level of sugar on the menu with above 120 sugar content
- Strawberry Shake (Large) and Chocolate Shake (Large) have sugar content of 123 and 120 respectively
- which means that the biggest way of getting sugar from the item is through McFlurry with M&M's Candies (Medium)

Checking for category with the highest and lowest level of Calories content on the menu

In [35]:
```
# Using groupby to the get the average content of calories on each category
Cal_level = Data.groupby('Category')['Calories'].mean().round().to_frame().sort_values
Cal_level
# plotting a barplot to display the average content of calories on each category of me
sns.barplot(y='Category', x='Calories', data=Cal_level, palette='Greys_r')
plt.title('Average level of Calories content based on Category') # Title label
```

Out[35]:
```
Text(0.5, 1.0, 'Average level of Calories content based on Category')
```



Observatios

- Chicken & Fish is the category we can get the highest amount of Calories
- Smoothies & Shakes and Breakfast have similar ways of get calories follow by Beef & Pork
- Beverages has the least way of getting Calories

Checking for the items with the highest level of calories on the menu

In [36]:
```
# grouping Item by Calories and getting the item with the maximium level of calories i
Cal_item = Data.groupby('Item')['Calories'].max().round().to_frame().sort_values('Calo
Cal_item
```

Out[36]:

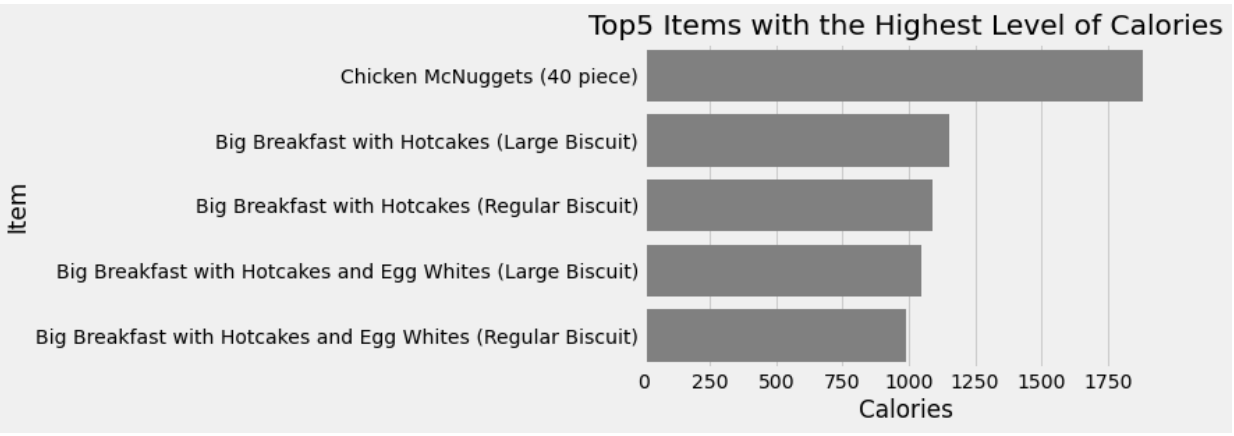
	Item	Calories
0	Chicken McNuggets (40 piece)	1880
1	Big Breakfast with Hotcakes (Large Biscuit)	1150
2	Big Breakfast with Hotcakes (Regular Biscuit)	1090
3	Big Breakfast with Hotcakes and Egg Whites (La...	1050
4	Big Breakfast with Hotcakes and Egg Whites (Re...	990

In [39]:
```
# Visualizing the top5 items with the highest level of calories
sns.barplot(y='Item', x='Calories', data= Cal_item, color = 'grey')

# chart title

plt.title('Top5 Items with the Highest Level of Calories')
```

Out[39]:
```
Text(0.5, 1.0, 'Top5 Items with the Highest Level of Calories')
```



Observations

- Chicken McNuggets (40 piece) is the item with the highest level of Calories
- Big Breakfast with Hotcakes (Large Biscuit) & Big Breakfast with Hotcakes (Regular Biscuit) are the items following ways of getting high Calories after Chicken McNuggets (40 piece).

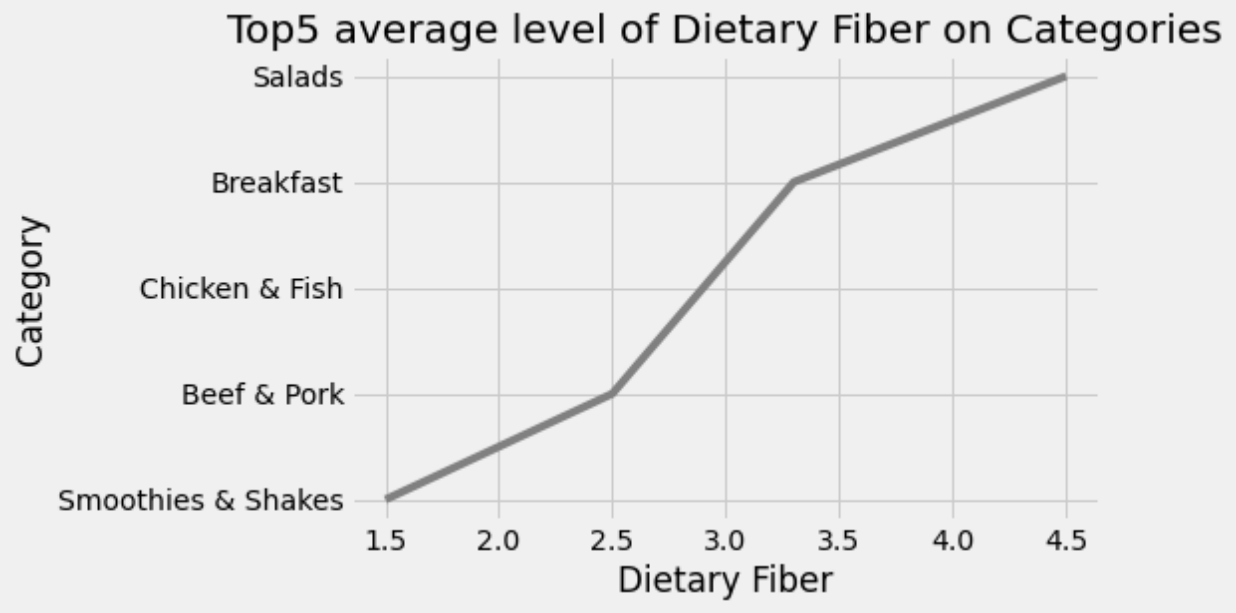
Checking for the category with the highest Fibre content on the menu

In [40]:
```
# using grouoby method to get the top5 average level of fibre content of the category
Fib = Data.groupby('Category')['Dietary Fiber'].mean().\
round(1).to_frame().sort_values('Dietary Fiber', ascending = False).head().reset_index

# visualizing the top5 average content of fibre on the category
sns.lineplot(y= 'Category', x='Dietary Fiber', data=Fib, color='grey')

# chart title
plt.title('Top5 average level of Dietary Fiber on Categories')
```

Out[40]:
```
Text(0.5, 1.0, 'Top5 average level of Dietary Fiber on Categories')
```



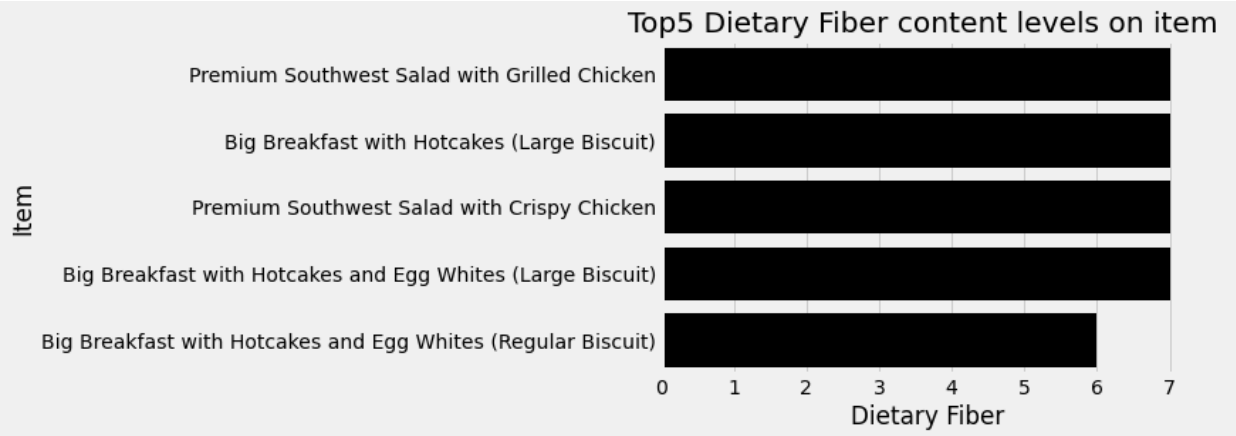
Observation

- From the graph we will noticed that Salads is the category with highest average level of Dietary Fiber on the menu
- Breakfast and Chicken&Fish respectively follows salads in a way of getting Dietary Fiber on the menu

Getting the items with high level of Dietary Fiber content on the menu

In [42]:
```
# Sorting out the top items with high level of Dietary Fiber using groupby method
fib_level=Data.groupby('Item')['Dietary Fiber'].max().round().to_frame().sort_values('
fib_level
# Visualizing the outcome of the top Dietary Fiber content on items
sns.barplot(y='Item', x='Dietary Fiber', data=fib_level, color = 'black' )
# Chart title
plt.title('Top5 Dietary Fiber content levels on item ')
```

Out[42]:
```
Text(0.5, 1.0, 'Top5 Dietary Fiber content levels on item ')
```



Observations

- The four items with same amount of Dietary Fiber content are; 1 Premium Southwest Salad with Grilled Chicken 2 Big Breakfast with Hotcakes (Large Biscuit)
  3 Premium Southwest Salad with Crispy Chicken
  4 Big Breakfast with Hotcakes and Egg Whites (Large Biscuit)

Conclusions

- From the overall analysis on McDonald's Nutrition Facts, its gattered that sodium is the nutrient with high content on the menu.
- There's an average level of about 3,600 content of sodium on the menu.
- The high content of sodium on the menu can be taking into consideration after the analysis shows that sodium is the nutrient present at high level, which is of controversial essence on the menu
- There are other nutrients with high level of content but sodium is of high consideration due to its controversial content on the menu

In [17]: