Milestone 1

In [1]:
```
import pandas as pd
```

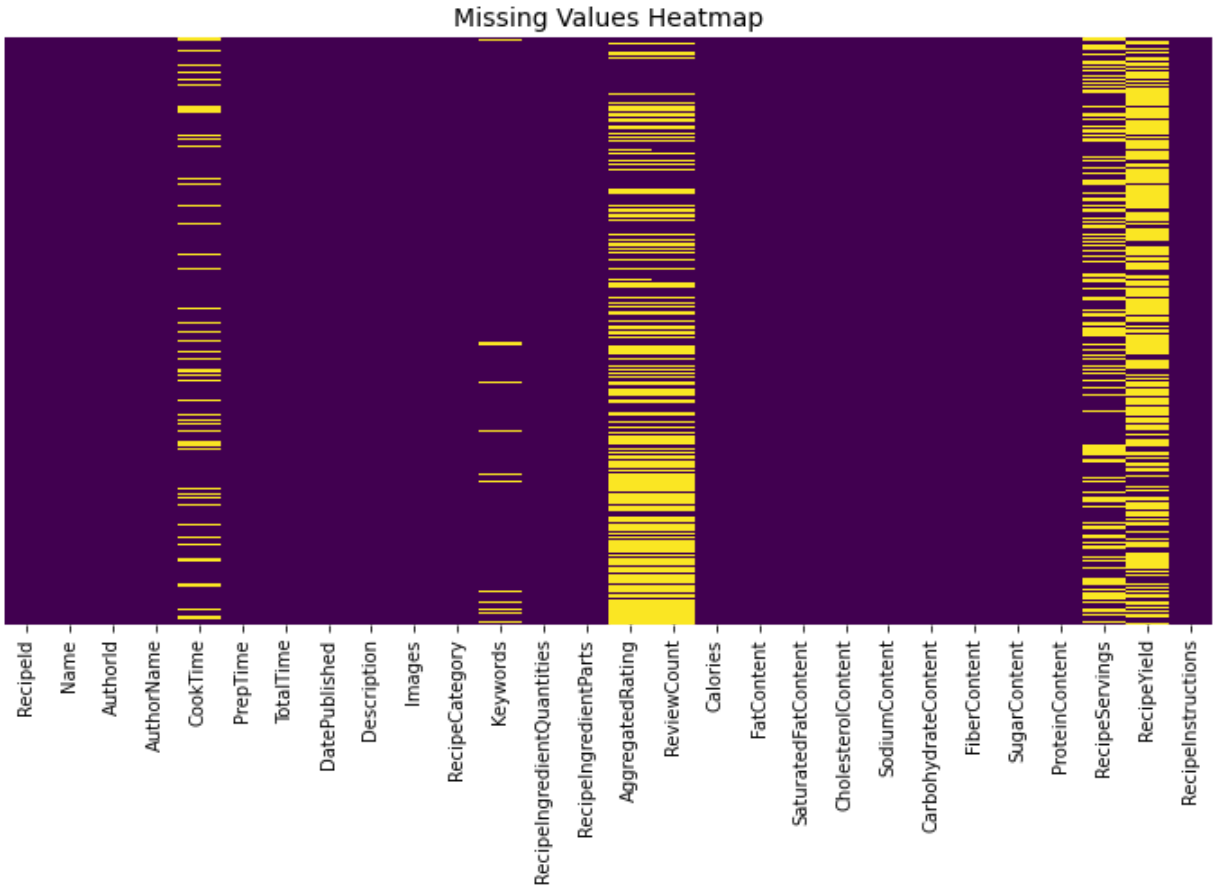
```
C:\Users\91705\anaconda3\lib\site-packages\pandas\core\computation\expressions.py:20: UserWarning: Pandas r
equires version '2.7.3' or newer of 'numexpr' (version '2.7.1' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

In [2]:
```
import pandas as pd

# Replace 'your_file.csv' with the path to your CSV file
df = pd.read_csv('recipes.csv')

# Display the first few rows of the dataframe
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522517 entries, 0 to 522516
Data columns (total 28 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   RecipeId                    522517 non-null  int64
 1   Name                        522517 non-null  object
 2   AuthorId                    522517 non-null  int64
 3   AuthorName                  522517 non-null  object
 4   CookTime                    439972 non-null  object
 5   PrepTime                    522517 non-null  object
 6   TotalTime                   522517 non-null  object
 7   DatePublished               522517 non-null  object
 8   Description                 522512 non-null  object
 9   Images                      522516 non-null  object
 10  RecipeCategory              521766 non-null  object
 11  Keywords                    505280 non-null  object
 12  RecipeIngredientQuantities  522514 non-null  object
 13  RecipeIngredientParts       522517 non-null  object
 14  AggregatedRating            269294 non-null  float64
 15  ReviewCount                 275028 non-null  float64
 16  Calories                    522517 non-null  float64
 17  FatContent                  522517 non-null  float64
 18  SaturatedFatContent         522517 non-null  float64
 19  CholesterolContent          522517 non-null  float64
 20  SodiumContent               522517 non-null  float64
 21  CarbohydrateContent         522517 non-null  float64
 22  FiberContent                522517 non-null  float64
 23  SugarContent                522517 non-null  float64
 24  ProteinContent              522517 non-null  float64
 25  RecipeServings              339606 non-null  float64
 26  RecipeYield                 174446 non-null  object
 27  RecipeInstructions          522517 non-null  object
dtypes: float64(12), int64(2), object(14)
memory usage: 111.6+ MB
```

In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset (replace with your actual dataset file)
df = pd.read_csv('recipes.csv')

# Set the figure size
plt.figure(figsize=(12, 6))

# Create a heatmap of missing values
sns.heatmap(df.isnull(), cmap="viridis", cbar=False, yticklabels=False)

# Add title
plt.title("Missing Values Heatmap", fontsize=14)

# Show plot
plt.show()
```



In [5]:

```
# Calculate the percentage of missing values per column
missing_percentage = (df.isnull().sum() / len(df)) * 100

# Convert to a DataFrame for better readability
missing_df = pd.DataFrame({"Column": df.columns, "Missing Percentage": missing_percentage})

# Display only columns with missing values
missing_df = missing_df[missing_df["Missing Percentage"] > 0].sort_values(by="Missing Percentage", ascendin

# Calculate the overall percentage of missing values
overall_missing_percentage = (df.isnull().sum().sum() / (df.shape[0] * df.shape[1])) * 100

# Print results
print("Percentage of Missing Values in Each Column:\n")
print(missing_df)
print("\nOverall Percentage of Missing Values in the Dataset: {:.2f}%".format(overall_missing_percentage))
```

```
Percentage of Missing Values in Each Column:

                                          Column  Missing Percentage
RecipeYield                           RecipeYield           66.614292
AggregatedRating                 AggregatedRating           48.462155
ReviewCount                           ReviewCount           47.364775
RecipeServings                     RecipeServings           35.005751
CookTime                                 CookTime           15.797572
Keywords                                 Keywords            3.298840
RecipeCategory                     RecipeCategory            0.143727
Description                           Description            0.000957
RecipeIngredientQuantities  RecipeIngredientQuantities            0.000574
Images                                     Images            0.000191

Overall Percentage of Missing Values in the Dataset: 7.74%
```

In [6]:

```
# Dropping the specified columns
columns_to_delete = [
    "Images",
    "PrepTime",
    "TotalTime",
    "DatePublished",
    "AggregatedRating",
    "ReviewCount",
    "RecipeServings",
    "RecipeYield",
    "AuthorId",
```

```
    "AuthorName",
    "RecipeIngredientQuantities",
    "SaturatedFatContent"
]

df = df.drop(columns=columns_to_delete)

# Verifying the updated dataframe structure
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522517 entries, 0 to 522516
Data columns (total 16 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   RecipeId               522517 non-null  int64  
 1   Name                   522517 non-null  object 
 2   CookTime               439972 non-null  object 
 3   Description            522512 non-null  object 
 4   RecipeCategory         521766 non-null  object 
 5   Keywords               505280 non-null  object 
 6   RecipeIngredientParts  522517 non-null  object 
 7   Calories               522517 non-null  float64
 8   FatContent             522517 non-null  float64
 9   CholesterolContent     522517 non-null  float64
 10  SodiumContent          522517 non-null  float64
 11  CarbohydrateContent    522517 non-null  float64
 12  FiberContent           522517 non-null  float64
 13  SugarContent           522517 non-null  float64
 14  ProteinContent         522517 non-null  float64
 15  RecipeInstructions     522517 non-null  object 
dtypes: float64(8), int64(1), object(7)
memory usage: 63.8+ MB
None
```

In [7]:

```
df = df.dropna(subset=['Description',])

# Display the updated DataFrame info
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 522512 entries, 0 to 522516
Data columns (total 16 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   RecipeId               522512 non-null  int64  
 1   Name                   522512 non-null  object 
 2   CookTime               439972 non-null  object 
 3   Description            522512 non-null  object 
 4   RecipeCategory         521761 non-null  object 
 5   Keywords               505275 non-null  object 
 6   RecipeIngredientParts  522512 non-null  object 
 7   Calories               522512 non-null  float64
 8   FatContent             522512 non-null  float64
 9   CholesterolContent     522512 non-null  float64
 10  SodiumContent          522512 non-null  float64
 11  CarbohydrateContent    522512 non-null  float64
 12  FiberContent           522512 non-null  float64
 13  SugarContent           522512 non-null  float64
 14  ProteinContent         522512 non-null  float64
 15  RecipeInstructions     522512 non-null  object 
dtypes: float64(8), int64(1), object(7)
memory usage: 67.8+ MB
None
```

In [8]:

```
# Drop rows where the 'Keywords' column is NaN
df = df.dropna(subset=['Keywords'])

# Display the updated DataFrame info
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 505275 entries, 0 to 522516
Data columns (total 16 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   RecipeId               505275 non-null  int64  
 1   Name                   505275 non-null  object 
 2   CookTime               424554 non-null  object 
 3   Description            505275 non-null  object 
```

```
 4   RecipeCategory          505275 non-null  object
 5   Keywords                505275 non-null  object
 6   RecipeIngredientParts   505275 non-null  object
 7   Calories                505275 non-null  float64
 8   FatContent              505275 non-null  float64
 9   CholesterolContent      505275 non-null  float64
 10  SodiumContent           505275 non-null  float64
 11  CarbohydrateContent     505275 non-null  float64
 12  FiberContent            505275 non-null  float64
 13  SugarContent            505275 non-null  float64
 14  ProteinContent          505275 non-null  float64
 15  RecipeInstructions      505275 non-null  object
dtypes: float64(8), int64(1), object(7)
memory usage: 65.5+ MB
None
```

In [9]:

```
# Convert the CookTime values to a timedelta object
df['CookTime'] = pd.to_timedelta(df['CookTime'], errors='coerce')
```

In [10]:

```
df.sample(5)
```

Out[10]:

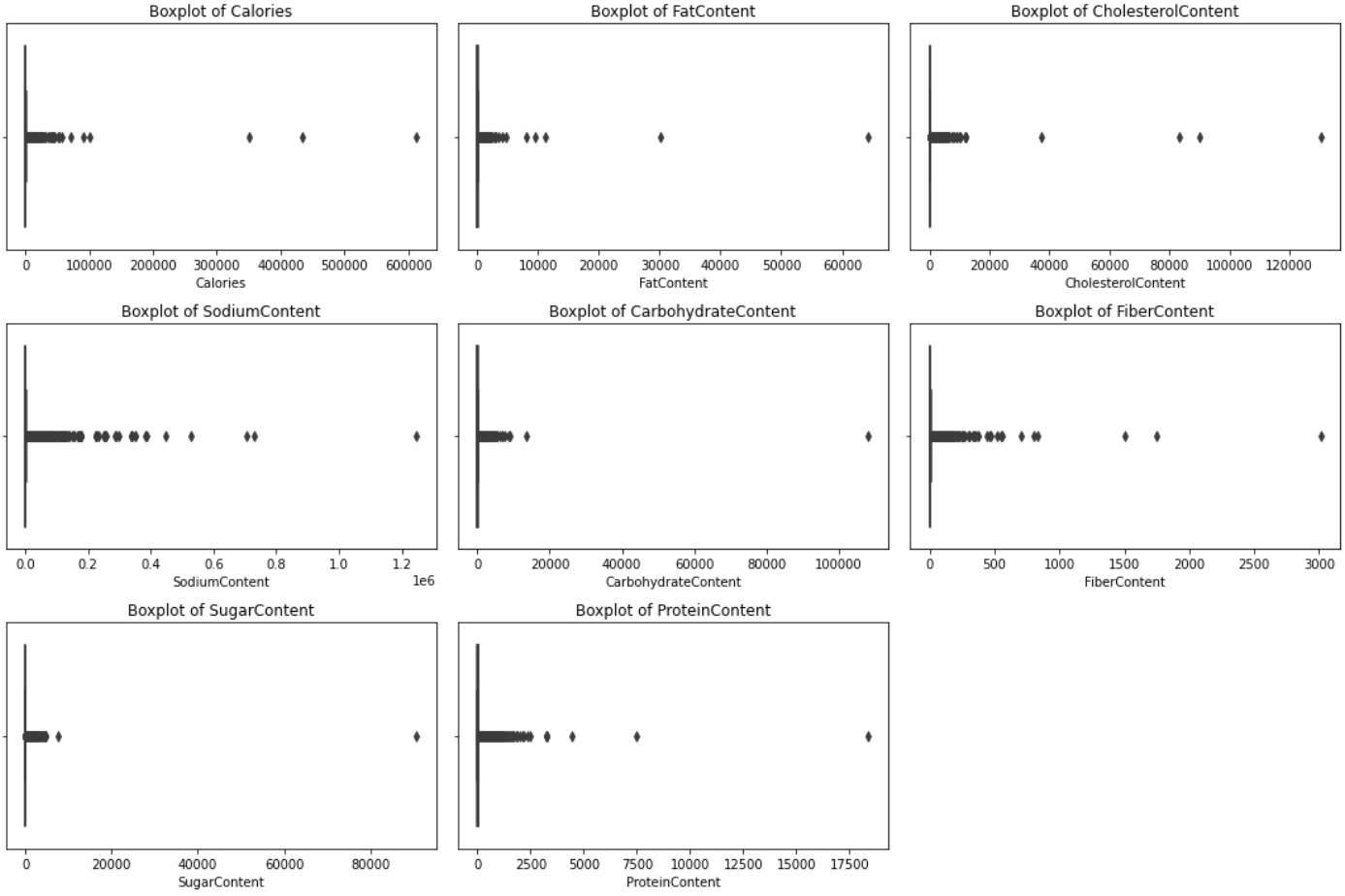
	RecipeId	Name	CookTime	Description	RecipeCategory	Keywords	RecipeIngredientParts	Calories	FatContent
119947	126202	Buttery Apricot Cookies	0 days 00:07:00	Make and share this Buttery Apricot Cookies re...	Dessert	"Cookie & Brownie"	c("butter", "powdered sugar", "vanilla", "egg"...	709.5	39.5
436319	452492	Chicken Salad Our Way!	0 days 01:00:00	My Mom and I came up with this one day while I...	Chicken	c("Poultry", "Meat", "< 4 Hours", "Easy")	c("light mayonnaise", "light sour cream", "pap...	224.7	14.1
485161	503055	Muffuletta Salad	NaT	Make and share this Muffuletta Salad recipe fr...	Lunch/Snacks	c("Cheese", "Greens", "Vegetable", "Meat", "Ca...	c("olive oil", "salt", "red wine vinegar", "ga...	540.1	34.1
326714	339064	Chamomile Lemonade	0 days 00:10:00	Make and share this Chamomile Lemonade recipe ...	Beverages	c("Lemon", "Citrus", "Fruit", "Summer", "< 15 ...	c("lemon zest", "lemon juice", "lemon slice")	105.3	0.0
20658	24080	Celery Victor	0 days 00:15:00	Something different as a side salad or an appe...	Lunch/Snacks	c("Vegetable", "Canadian", "Free Of...", "< 30...	c("celery hearts", "celery", "wine vinegar", "...	8.7	0.1

In [11]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming df is your DataFrame
columns_of_interest = [
    'Calories', 'FatContent',
    'CholesterolContent', 'SodiumContent', 'CarbohydrateContent',
    'FiberContent', 'SugarContent', 'ProteinContent'
]

# Calculate statistical details
stats = df[columns_of_interest].describe(percentiles=[.25, .5, .75])

# Display the statistics
print(stats)

# Create boxplots for each column to visualize distributions
plt.figure(figsize=(15, 10))
for i, col in enumerate(columns_of_interest, 1):
    plt.subplot(3, 3, i)
```

```
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
plt.tight_layout()
plt.show()
```

```
           Calories     FatContent  CholesterolContent  SodiumContent  \
count  505275.000000  505275.000000       505275.000000   5.052750e+05
mean      483.633761      24.515018           86.040845   7.582017e+02
std      1415.194844     113.054447          305.790997   4.166146e+03
min         0.000000       0.000000            0.000000   0.000000e+00
25%       172.700000       5.500000            3.600000   1.205000e+02
50%       315.100000      13.600000           41.700000   3.471000e+02
75%       526.800000      27.250000          106.900000   7.835000e+02
max    612854.600000   64368.100000       130456.400000   1.246921e+06

       CarbohydrateContent   FiberContent   SugarContent  ProteinContent
count        505275.000000  505275.000000  505275.000000   505275.000000
mean             49.296306       3.823619      22.140023       17.301844
std             183.212983       8.653774     144.550681       40.480393
min               0.000000       0.000000       0.000000        0.000000
25%              12.700000       0.800000       2.500000        3.400000
50%              28.200000       2.100000       6.500000        8.900000
75%              51.100000       4.500000      18.200000       24.800000
max          108294.600000    3012.000000   90682.300000    18396.200000
```

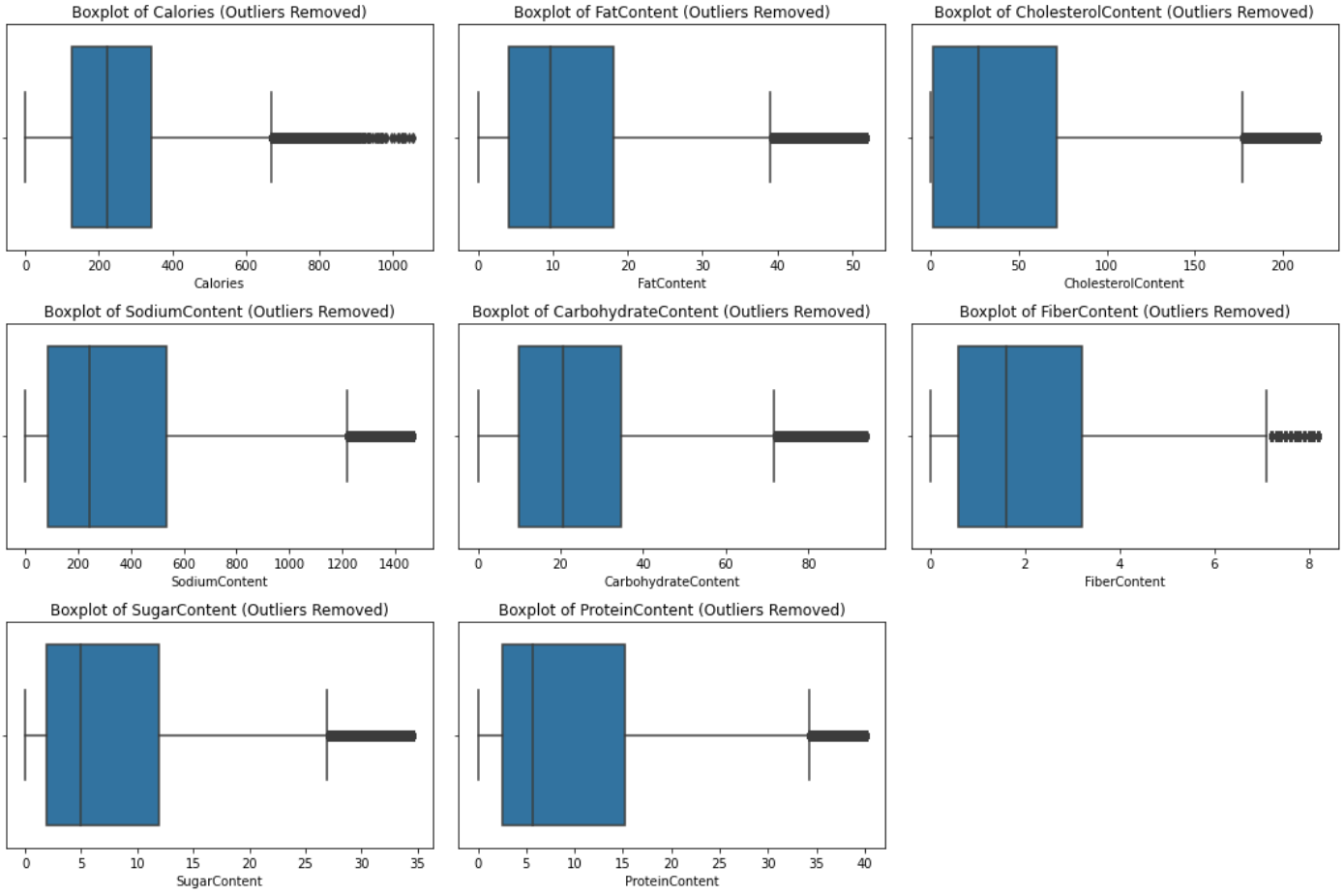


In [12]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Define columns to check
columns_of_interest = [
    'Calories', 'FatContent',
    'CholesterolContent', 'SodiumContent', 'CarbohydrateContent',
    'FiberContent', 'SugarContent', 'ProteinContent'
]

# Function to remove outliers based on IQR
def remove_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df
```

```
# Remove outliers from the dataset
df_no_outliers = remove_outliers(df, columns_of_interest)

# Calculate the new statistics after removing outliers
stats_no_outliers = df_no_outliers[columns_of_interest].describe(percentiles=[.25, .5, .75])

# Display the new statistics
print(stats_no_outliers)

# Create boxplots for each column after removing outliers
plt.figure(figsize=(15, 10))
for i, col in enumerate(columns_of_interest, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(x=df_no_outliers[col])
    plt.title(f'Boxplot of {col} (Outliers Removed)')
plt.tight_layout()
plt.show()
```

```
            Calories     FatContent  CholesterolContent  SodiumContent  \
count  327485.000000  327485.000000       327485.000000  327485.000000
mean      247.340458      12.391694           43.849851     354.414903
std       155.040509      10.672985           48.626822     340.035898
min         0.000000       0.000000            0.000000       0.000000
25%       127.400000       4.100000            1.200000      84.200000
50%       222.500000       9.700000           27.100000     244.400000
75%       343.600000      18.100000           71.600000     537.400000
max      1054.300000      51.800000          220.200000    1468.700000

       CarbohydrateContent   FiberContent   SugarContent  ProteinContent
count        327485.000000  327485.000000  327485.000000   327485.000000
mean             23.967280       2.164405       8.182250       10.058629
std              17.798222       1.933437       8.482603       10.247189
min               0.000000       0.000000       0.000000        0.000000
25%               9.800000       0.600000       1.900000        2.500000
50%              20.700000       1.600000       4.900000        5.700000
75%              34.600000       3.200000      11.900000       15.200000
max              93.900000       8.200000      34.600000       40.100000
```



In [13]:

```
df_no_outliers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 327485 entries, 0 to 522515
Data columns (total 16 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
```

```
 0   RecipeId                327485 non-null  int64
 1   Name                    327485 non-null  object
 2   CookTime                270755 non-null  timedelta64[ns]
 3   Description             327485 non-null  object
 4   RecipeCategory          327485 non-null  object
 5   Keywords                327485 non-null  object
 6   RecipeIngredientParts   327485 non-null  object
 7   Calories                327485 non-null  float64
 8   FatContent              327485 non-null  float64
 9   CholesterolContent      327485 non-null  float64
 10  SodiumContent           327485 non-null  float64
 11  CarbohydrateContent     327485 non-null  float64
 12  FiberContent            327485 non-null  float64
 13  SugarContent            327485 non-null  float64
 14  ProteinContent          327485 non-null  float64
 15  RecipeInstructions      327485 non-null  object
dtypes: float64(8), int64(1), object(6), timedelta64[ns](1)
memory usage: 42.5+ MB
```

In [14]:

```
# Save the cleaned dataset as a CSV file in the current directory
df_no_outliers.to_csv("cleaned_recipes_dataset.csv", index=False)

# Confirm the save was successful
print("Cleaned dataset saved as 'cleaned_recipes_dataset.csv' in the current directory.")
```

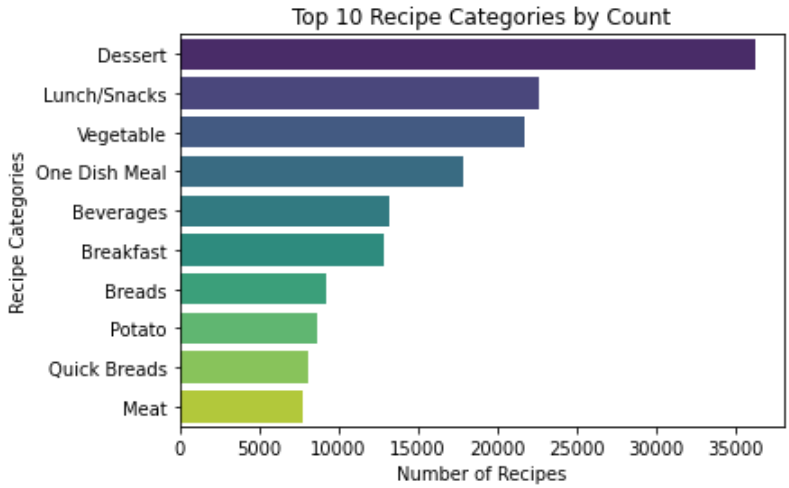
```
Cleaned dataset saved as 'cleaned_recipes_dataset.csv' in the current directory.
```

In [ ]:

EDA of the Cleaned Dataset

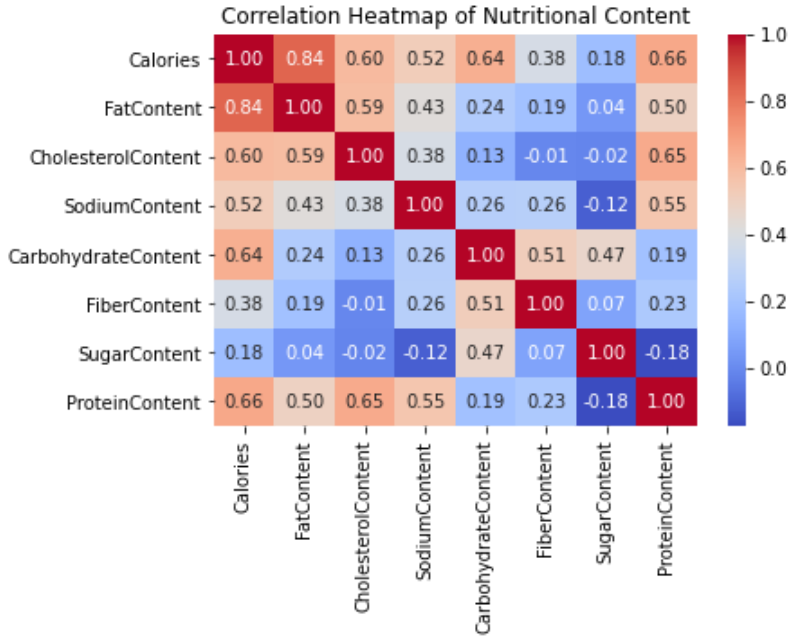
In [15]:

```
import matplotlib.pyplot as plt
import seaborn as sns

category_counts = df_no_outliers['RecipeCategory'].value_counts().head(10)
sns.barplot(x=category_counts.values, y=category_counts.index, palette="viridis")
plt.title("Top 10 Recipe Categories by Count")
plt.xlabel("Number of Recipes")
plt.ylabel("Recipe Categories")
plt.show()
```



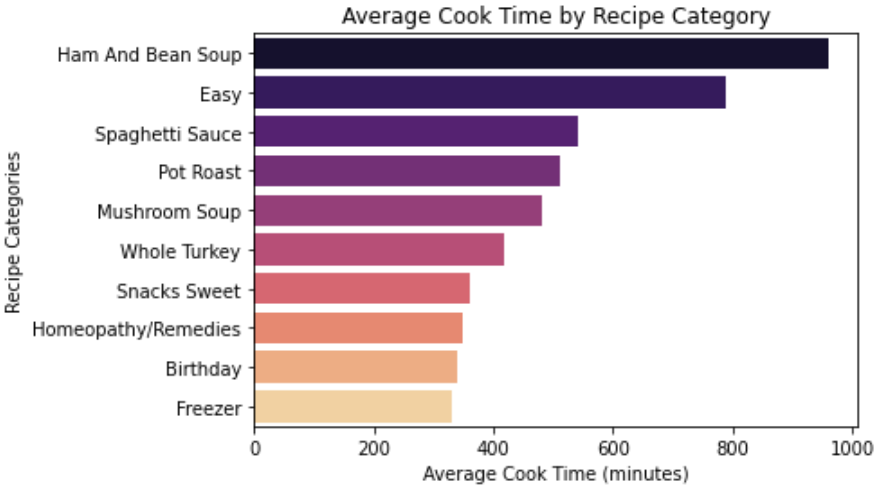
In [16]:

```
numeric_columns = ['Calories', 'FatContent', 'CholesterolContent', 'SodiumContent',
                   'CarbohydrateContent', 'FiberContent', 'SugarContent', 'ProteinContent']

correlation_matrix = df_no_outliers[numeric_columns].corr()

sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap of Nutritional Content")
plt.show()
```



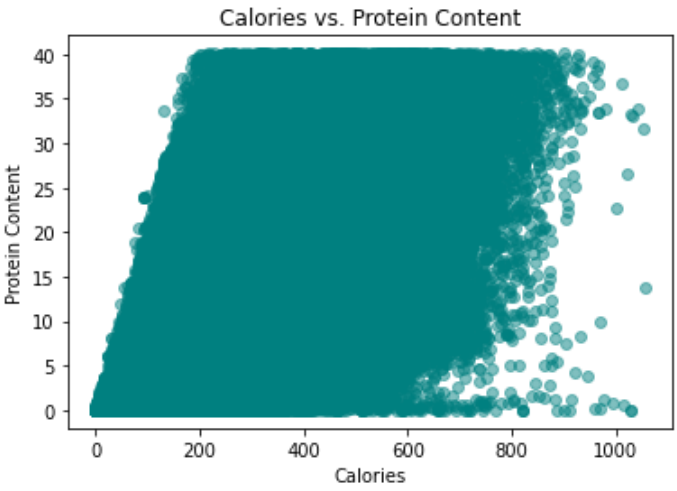
In [17]:

```
avg_cook_time = df_no_outliers.groupby('RecipeCategory')['CookTime'].mean().sort_values(ascending=False).he
avg_cook_time = avg_cook_time.dt.total_seconds() / 60  # Convert to minutes

sns.barplot(x=avg_cook_time.values, y=avg_cook_time.index, palette="magma")
plt.title("Average Cook Time by Recipe Category")
plt.xlabel("Average Cook Time (minutes)")
plt.ylabel("Recipe Categories")
plt.show()
```



In [18]:

```
plt.scatter(df_no_outliers['Calories'], df_no_outliers['ProteinContent'], alpha=0.5, color="teal")
plt.title("Calories vs. Protein Content")
plt.xlabel("Calories")
plt.ylabel("Protein Content")
plt.show()
```

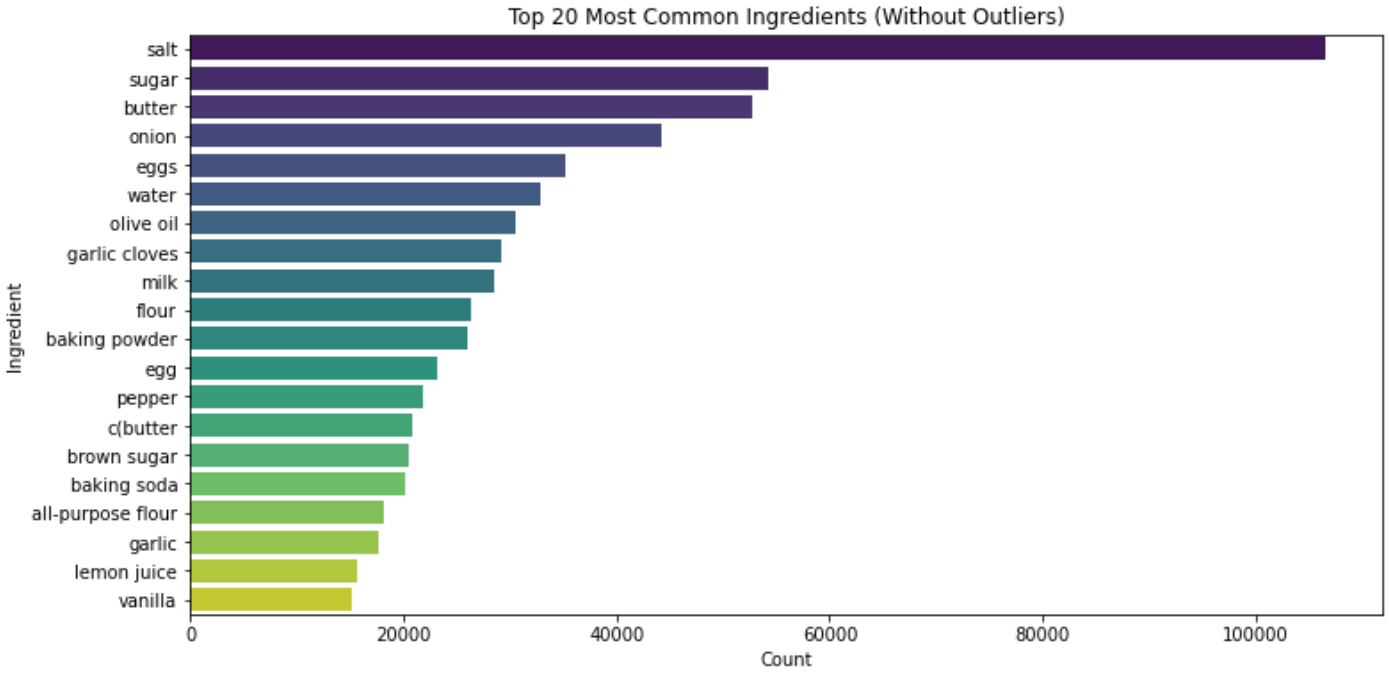


In [19]:
```
df_no_outliers.sample(5)
```

Out[19]:

	RecipeId	Name	CookTime	Description	RecipeCategory	Keywords	RecipeIngredientParts	Calories	FatContent
511181	529707	Breakfast Quinoa	0 days 03:00:00	Slow cooker breakfast option. You can cook t...	Breakfast	c("Low Cholesterol", "< 4 Hours", "Easy")	c("quinoa", "milk", "dates", "cinnamon", "nutm...	295.4	10.7
336074	348708	Georgetown Cupcake's Chocolate Ganache Cupcakes	0 days 00:20:00	This was the winning recipe in the 8 week long...	Dessert	c("< 60 Mins", "For Large Groups")	c("flour", "baking soda", "salt", "unsalted bu...	220.7	11.8
399399	414030	Clare's Chocolate-Walnut-Espresso Brownies	0 days 00:40:00	Make and share this Clare's Chocolate-Walnut-E...	Dessert	c("Cookie & Brownie", "< 60 Mins", "For Large ...	c("instant espresso", "walnuts", "vanilla")	210.6	10.8
270000	280767	Grilled Italian Tomatoes	0 days 00:50:00	Make and share this Grilled Italian Tomatoes r...	Vegetable	c("Low Protein", "Low Cholesterol", "Oven", "<...	c("Italian tomatoes", "olive oil", "salt", "fr...	41.4	3.5
403562	418341	Ww Greek-Style Spaghetti Squash (Works W/Simpl...	0 days 00:30:00	I got this from my WW leader. Although I'm not...	Low Protein	c("Low Cholesterol", "Healthy", "< 30 Mins")	c("spaghetti squash", "extra virgin olive oil"...	212.1	6.5

In [20]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from itertools import combinations
import networkx as nx


# Split the ingredient lists (assuming they are stored as comma-separated values)
df_no_outliers["RecipeIngredientParts"] = df_no_outliers["RecipeIngredientParts"].astype(str).apply(lambda

# Flatten the list of all ingredients
all_ingredients = [ingredient.strip().lower() for ingredients in df_no_outliers["RecipeIngredientParts"] fo

# Count the frequency of each ingredient
ingredient_counts = Counter(all_ingredients)

# Convert to DataFrame for visualization
ingredient_df = pd.DataFrame(ingredient_counts.items(), columns=["Ingredient", "Count"]).sort_values(by="Co

# Plot the most common ingredients
plt.figure(figsize=(12, 6))
sns.barplot(x=ingredient_df["Count"], y=ingredient_df["Ingredient"], palette="viridis")
plt.title("Top 20 Most Common Ingredients (Without Outliers)")
plt.xlabel("Count")
plt.ylabel("Ingredient")
plt.show()
```



Milestone 2

In [21]:

```
df_no_outliers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 327485 entries, 0 to 522515
Data columns (total 16 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   RecipeId               327485 non-null  int64
 1   Name                   327485 non-null  object
 2   CookTime               270755 non-null  timedelta64[ns]
 3   Description            327485 non-null  object
 4   RecipeCategory         327485 non-null  object
 5   Keywords               327485 non-null  object
 6   RecipeIngredientParts  327485 non-null  object
 7   Calories               327485 non-null  float64
 8   FatContent             327485 non-null  float64
 9   CholesterolContent     327485 non-null  float64
 10  SodiumContent          327485 non-null  float64
 11  CarbohydrateContent    327485 non-null  float64
 12  FiberContent           327485 non-null  float64
 13  SugarContent           327485 non-null  float64
 14  ProteinContent         327485 non-null  float64
 15  RecipeInstructions     327485 non-null  object
dtypes: float64(8), int64(1), object(6), timedelta64[ns](1)
memory usage: 42.5+ MB
```

In [22]:

```
# Define the columns to process
columns_of_interest = [
    'Calories', 'FatContent', 'CholesterolContent',
    'SodiumContent', 'CarbohydrateContent', 'FiberContent',
    'SugarContent', 'ProteinContent'
]

# Function to categorize content level
def categorize_level(value, Q1, Q2, Q3):
    # Apply conditions based on quartiles
    if value <= Q1:
        return 1  # Very low
    elif value <= Q2:
        return 2  # Low
    elif value <= Q3:
        return 3  # Medium
    else:
        return 4  # High

# Calculate the quartiles for each column of interest
quartiles = {}
for col in columns_of_interest:
    Q1 = df_no_outliers[col].quantile(0.25)
```

```
    Q2 = df_no_outliers[col].quantile(0.50)
    Q3 = df_no_outliers[col].quantile(0.75)
    quartiles[col] = (Q1, Q2, Q3)

# Apply the categorization to each column and create new columns
for col in columns_of_interest:
    Q1, Q2, Q3 = quartiles[col]
    df_no_outliers[f'{col}_Level'] = df_no_outliers[col].apply(categorize_level, args=(Q1, Q2, Q3))

# Show a sample of the new columns
df_no_outliers[['Calories', 'Calories_Level', 'FatContent', 'FatContent_Level','CholesterolContent','Choles
```

Out[22]:

	Calories	Calories_Level	FatContent	FatContent_Level	CholesterolContent	CholesterolContent_Level	SodiumContent	Sc
20468	120.6	1	2.6	1	8.2	2	79.2	
380377	400.1	4	30.3	4	46.5	3	149.9	
133643	206.3	2	11.6	3	0.0	1	308.8	
70177	325.5	3	15.9	3	53.4	3	616.9	
478418	223.5	3	16.5	3	35.9	3	1263.8	
388562	357.8	4	29.3	4	104.2	4	547.2	
433769	312.5	3	4.1	1	0.0	1	803.7	
206410	379.7	4	30.4	4	88.6	4	569.7	
129441	495.8	4	34.1	4	122.5	4	149.9	
500074	291.9	3	14.5	3	25.7	2	168.1	

In [23]:

```
df_no_outliers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 327485 entries, 0 to 522515
Data columns (total 24 columns):
 #   Column                     Non-Null Count   Dtype          
---  ------                     --------------   -----          
 0   RecipeId                   327485 non-null  int64          
 1   Name                       327485 non-null  object         
 2   CookTime                   270755 non-null  timedelta64[ns]
 3   Description                327485 non-null  object         
 4   RecipeCategory             327485 non-null  object         
 5   Keywords                   327485 non-null  object         
 6   RecipeIngredientParts      327485 non-null  object         
 7   Calories                   327485 non-null  float64        
 8   FatContent                 327485 non-null  float64        
 9   CholesterolContent         327485 non-null  float64        
 10  SodiumContent              327485 non-null  float64        
 11  CarbohydrateContent        327485 non-null  float64        
 12  FiberContent               327485 non-null  float64        
 13  SugarContent               327485 non-null  float64        
 14  ProteinContent             327485 non-null  float64        
 15  RecipeInstructions         327485 non-null  object         
 16  Calories_Level             327485 non-null  int64          
 17  FatContent_Level           327485 non-null  int64          
 18  CholesterolContent_Level   327485 non-null  int64          
 19  SodiumContent_Level        327485 non-null  int64          
 20  CarbohydrateContent_Level  327485 non-null  int64          
 21  FiberContent_Level         327485 non-null  int64          
 22  SugarContent_Level         327485 non-null  int64          
 23  ProteinContent_Level       327485 non-null  int64          
dtypes: float64(8), int64(9), object(6), timedelta64[ns](1)
memory usage: 62.5+ MB
```

In [24]:

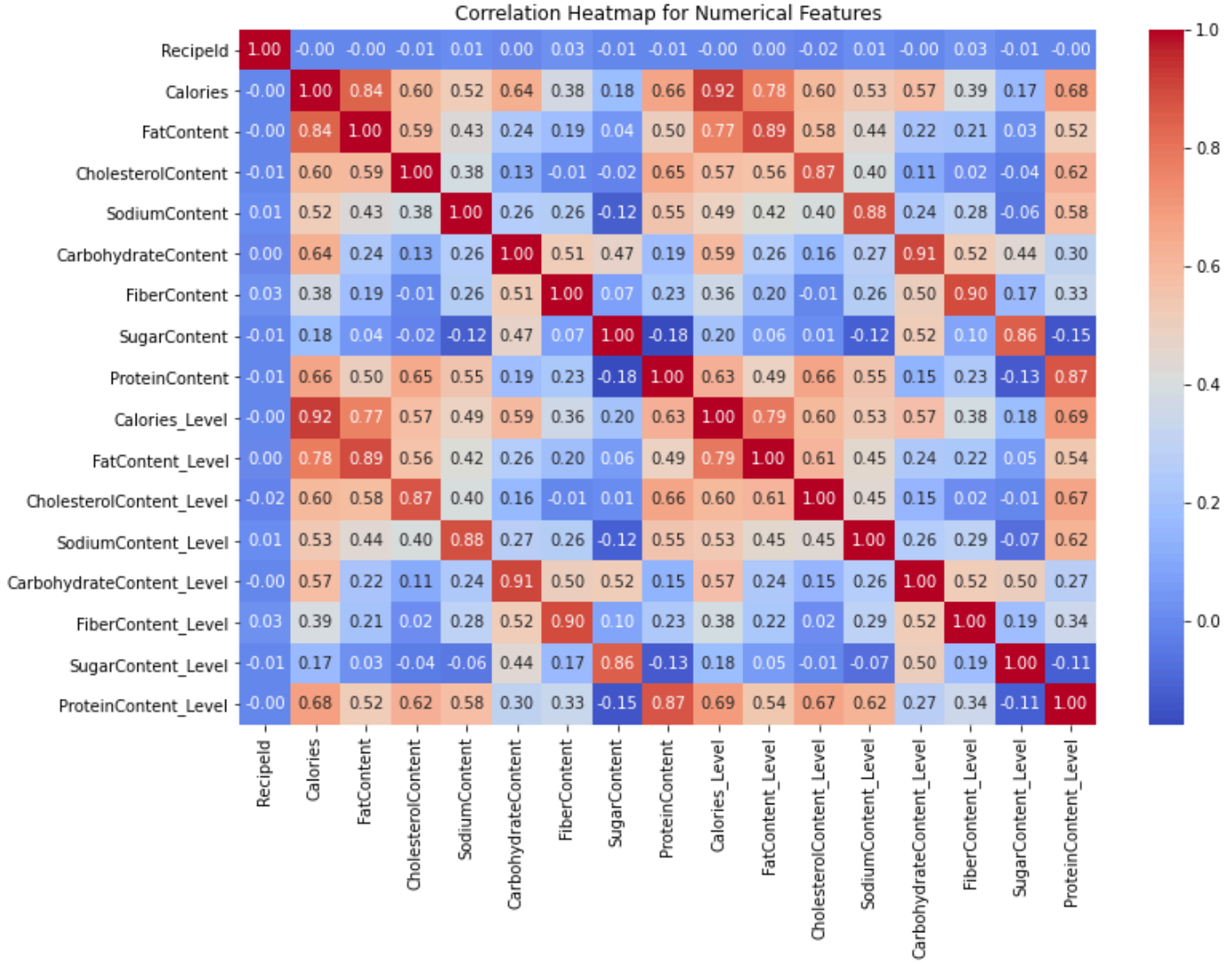
```
# Save the cleaned dataset as a CSV file in the current directory
df_no_outliers.to_csv("cleaned_recipes_dataset.csv", index=False)

# Confirm the save was successful
print("Cleaned dataset saved as 'cleaned_recipes_dataset.csv' in the current directory.")
```

```
Cleaned dataset saved as 'cleaned_recipes_dataset.csv' in the current directory.
```

Feature Selection

In [26]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Select only numerical columns
numerical_cols = df_no_outliers.select_dtypes(include=['float64', 'int64']).columns

# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df_no_outliers[numerical_cols].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap for Numerical Features")
plt.show()
```



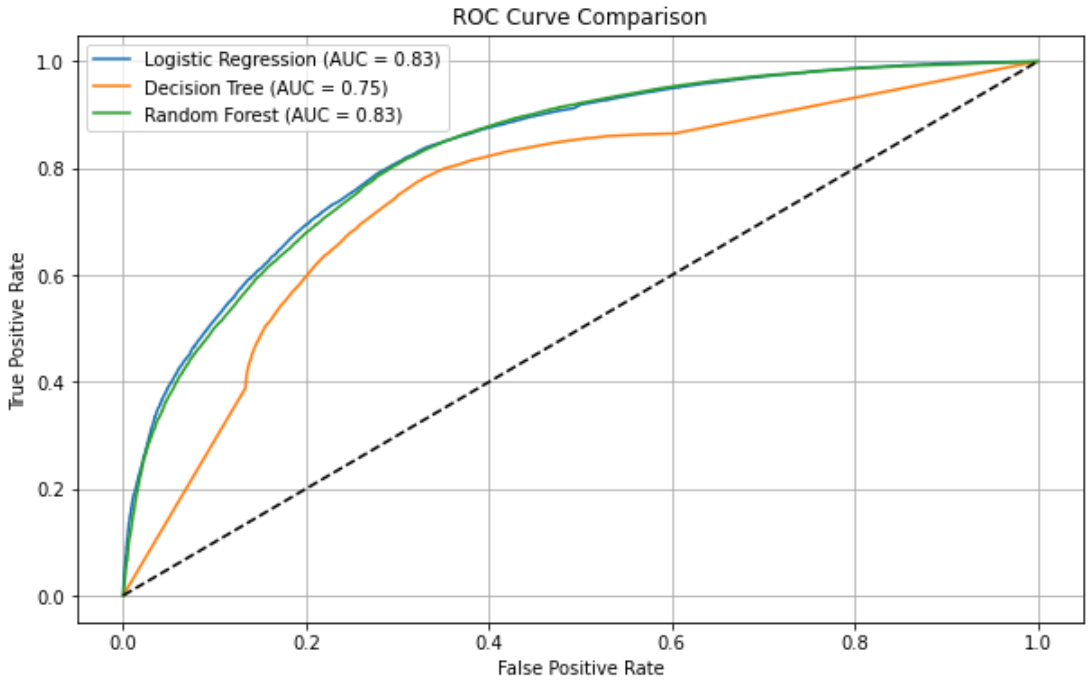
In [ ]:

```
```

In [27]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score, roc_curve
import matplotlib.pyplot as plt

# Assuming df_modelling is already available in the environment
df_modelling = df_no_outliers.dropna(subset=['Keywords', 'ProteinContent_Level'])
df_modelling['HighProtein'] = df_modelling['ProteinContent_Level'].apply(lambda x: 1 if x >= 3 else 0)

# Feature and target
X = df_modelling['Keywords'].astype(str)
y = df_modelling['HighProtein']
```

```python
# Vectorization
vectorizer = TfidfVectorizer(max_features=1000)
X_vec = vectorizer.fit_transform(X)

# Split into train, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X_vec, y, test_size=0.3, random_state=42, stratify=y)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_

# Model initialization
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier()
}

results = {}

# Train and evaluate models
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_val)
    y_proba = model.predict_proba(X_val)[:, 1]
    report = classification_report(y_val, y_pred, output_dict=True)
    auc = roc_auc_score(y_val, y_proba)
    fpr, tpr, _ = roc_curve(y_val, y_proba)
    results[name] = {
        "model": model,
        "report": report,
        "auc": auc,
        "fpr": fpr,
        "tpr": tpr
    }

# Plot ROC Curves
plt.figure(figsize=(10, 6))
for name, res in results.items():
    plt.plot(res["fpr"], res["tpr"], label=f"{name} (AUC = {res['auc']:.2f})")
plt.plot([0, 1], [0, 1], 'k--')
plt.title("ROC Curve Comparison")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.grid(True)
plt.show()

# Display classification report summaries
summary = {
    model: {
        "Accuracy": res["report"]["accuracy"],
        "Precision": res["report"]["1"]["precision"],
        "Recall": res["report"]["1"]["recall"],
        "F1-Score": res["report"]["1"]["f1-score"],
        "AUC": res["auc"]
    }
    for model, res in results.items()
}
```



In [33]:

```
# Rebuild the summary dictionary
summary = {
    model: {
        "Accuracy": res["report"]["accuracy"],
        "Precision": res["report"]["1"]["precision"],
        "Recall": res["report"]["1"]["recall"],
        "F1-Score": res["report"]["1"]["f1-score"],
        "AUC": res["auc"]
    }
    for model, res in results.items()
}

# Convert to DataFrame
summary_df = pd.DataFrame(summary).T

# Display in Jupyter
from IPython.display import display
display(summary_df)
```

	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.755634	0.735691	0.795433	0.764396	0.833654
Decision Tree	0.724447	0.712500	0.749520	0.730541	0.750857
Random Forest	0.753435	0.735161	0.789755	0.761481	0.830269

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: