```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('final.csv')
df.head()
```

```
                                          directions   fat  \
0  ['1. Place the stock, lentils, celery, carrot,...   7.0
1  ['Combine first 9 ingredients in heavy medium ...  23.0
2  ['In a large heavy saucepan cook diced fennel ...   7.0
3  ['Heat oil in heavy large skillet over medium-...   NaN
4  ['Preheat oven to 350°F. Lightly grease 8x8x2-...  32.0

                       date  \
0  2006-09-01T04:00:00.000Z
1  2004-08-20T04:00:00.000Z
2  2004-08-20T04:00:00.000Z
3  2009-03-27T04:00:00.000Z
4  2004-08-20T04:00:00.000Z

                                          categories  calories  \
0  ['Sandwich', 'Bean', 'Fruit', 'Tomato', 'turke...     426.0
1  ['Food Processor', 'Onion', 'Pork', 'Bake', 'B...     403.0
2  ['Soup/Stew', 'Dairy', 'Potato', 'Vegetable', ...     165.0
3  ['Fish', 'Olive', 'Tomato', 'Sauté', 'Low Fat'...       NaN
4  ['Cheese', 'Dairy', 'Pasta', 'Vegetable', 'Sid...     547.0

                                                desc  protein  rating
\
0                                                NaN     30.0   2.500

1  This uses the same ingredients found in boudin...     18.0   4.375

2                                                NaN      6.0   3.750

3  The Sicilian-style tomato sauce has tons of Me...      NaN   5.000

4                                                NaN     20.0   3.125


                                        title  \
0               Lentil, Apple, and Turkey Wrap
1  Boudin Blanc Terrine with Red Onion Confit
2                    Potato and Fennel Soup Hodge
3             Mahi-Mahi in Tomato Olive Sauce
4                     Spinach Noodle Casserole
```

```
                                         ingredients  sodium
0  ['4 cups low-sodium vegetable or chicken stock...   559.0
1  ['1 1/2 cups whipping cream', '2 medium onions...  1439.0
2  ['1 fennel bulb (sometimes called anise), stal...   165.0
3  ['2 tablespoons extra-virgin olive oil', '1 cu...     NaN
4  ['1 12-ounce package frozen spinach soufflé, t...   452.0
```

removing unnecessay columns

```
columns_to_remove = ['desc','title']
df.drop(columns_to_remove,axis = 1,inplace = True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20130 entries, 0 to 20129
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   directions   20111 non-null  object 
 1   fat          15908 non-null  float64
 2   date         20111 non-null  object 
 3   categories   20111 non-null  object 
 4   calories     15976 non-null  float64
 5   protein      15929 non-null  float64
 6   rating       20100 non-null  float64
 7   ingredients  20111 non-null  object 
 8   sodium       15974 non-null  float64
dtypes: float64(5), object(4)
memory usage: 1.4+ MB
```

checking the null values

```
df.isnull().sum()
```

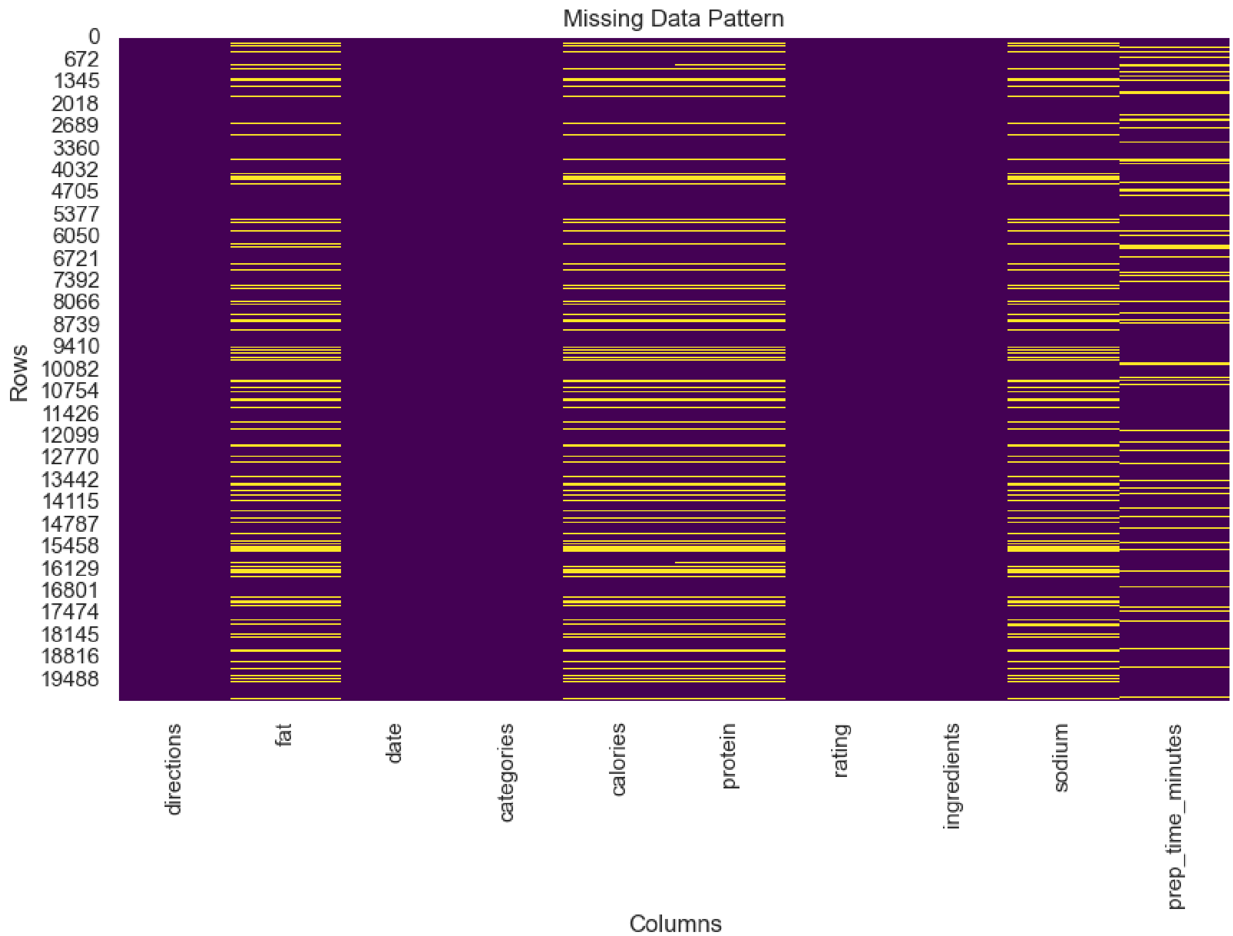
```
directions       19
fat            4222
date             19
categories       19
calories       4154
protein        4201
rating           30
ingredients      19
sodium         4156
dtype: int64
```

```
null = df.isnull().mean()*100
perc = null.apply(lambda x:f"{x : .2f}%")
print(perc)
```

```
directions       0.09%
fat             20.97%
date             0.09%
categories       0.09%
calories        20.64%
protein         20.87%
rating           0.15%
ingredients      0.09%
sodium          20.65%
dtype: object
```

dropping null values from ratings

```python
df.dropna(subset = ['rating'],inplace = True)

# Analyzing missing data patterns
plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Data Pattern')
plt.xlabel('Columns')
plt.ylabel('Rows')
plt.show()
```




```
df.describe()
```

	fat	calories	protein	rating	sodium
count	1.590100e+04	1.596900e+04	15922.000000	20100.000000	1.596700e+04
mean	3.462407e+02	6.310443e+03	99.982665	3.713060	6.213949e+03
std	2.043552e+04	3.586637e+05	3836.459371	1.343144	3.329632e+05
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000e+00
25%	7.000000e+00	1.980000e+02	3.000000	3.750000	8.000000e+01
50%	1.700000e+01	3.310000e+02	8.000000	4.375000	2.940000e+02
75%	3.300000e+01	5.860000e+02	27.000000	4.375000	7.110000e+02
max	1.722763e+06	3.011122e+07	236489.000000	5.000000	2.767511e+07

Distribution: The wide range between the min and max values, along with the standard deviation, suggests significant variability in the data.

Skewness: If the mean is higher than the median, it could indicate a right skew in the distribution. Conversely, if the median is higher than the mean, it might indicate a left skew.

Potential Outliers: The max values being significantly higher than the 75th percentile may indicate the presence of outliers in the dataset, especially in columns with large standard deviations.

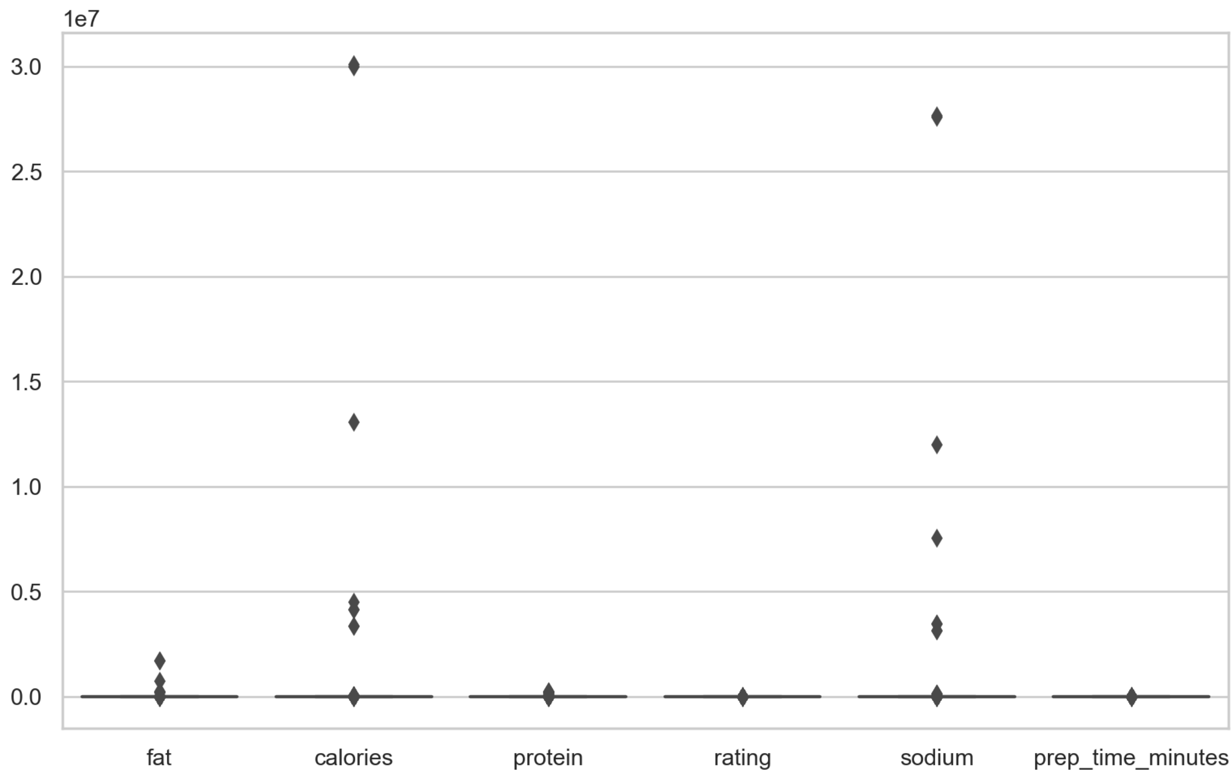
checking outliers

```
def identify_outlier_indices_iqr(df):
    outlier_indices = set()
    for column in df.select_dtypes(include=[np.number]).columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR


        outlier_indices.update(df[(df[column] < lower_bound) |
(df[column] > upper_bound)].index)

    return list(outlier_indices)

# Identify outlier indices
outlier_indices = identify_outlier_indices_iqr(df)
print(f"Number of outlier rows: {len(outlier_indices)}")
```

```
Number of outlier rows: 5666
```

```
plt.figure(figsize= (10,6),dpi = 200)
sns.boxplot(df)
```

```
<Axes: >
```



Since we are specifically interested in finding the common ingredients in the top 10 highly-rated recipes and preptime and rating relationship, removing outliers from those columns might not be necessary or even helpful. Here's why:

Focus on Ingredients: our goal is to analyze the ingredients of the top-rated recipes, and the nutritional values (calories, proteins, fat, sodium) are secondary. The ingredients themselves are unlikely to be "outliers" in the same sense as numerical columns.

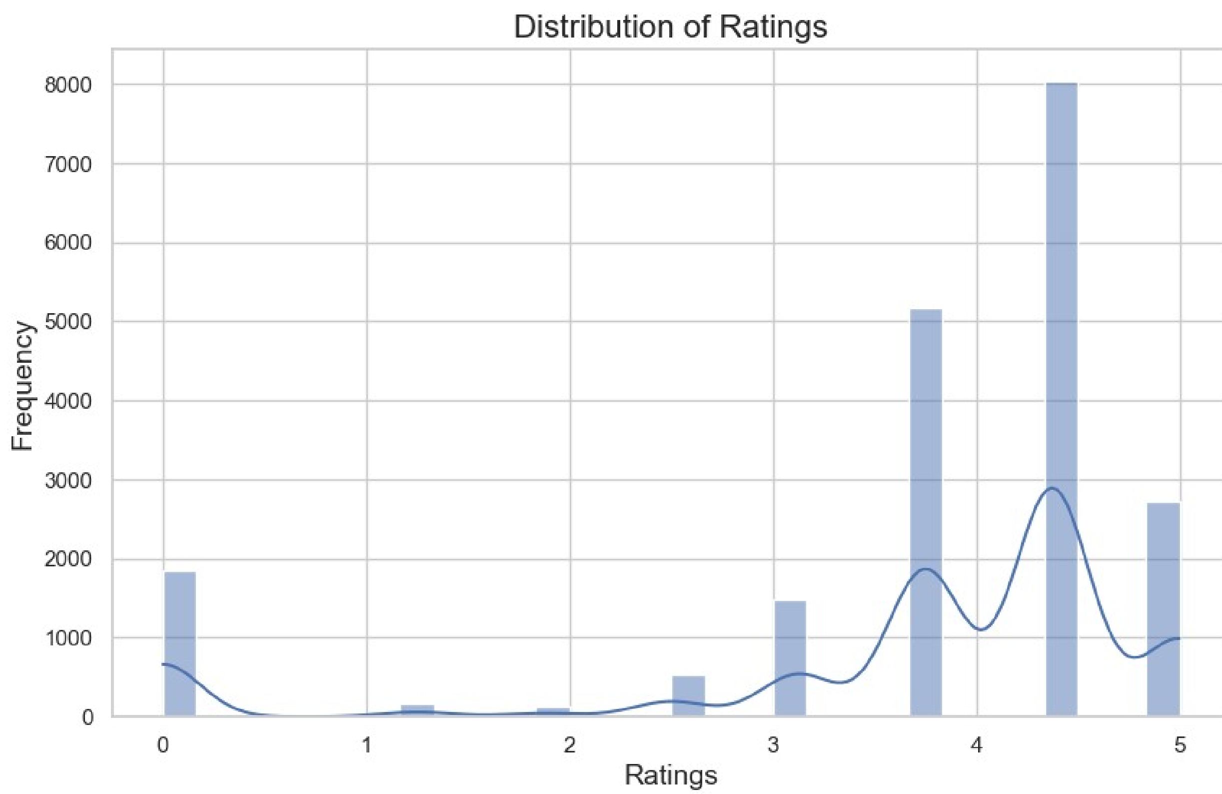
Preserving Recipe Context: Outliers in columns like calories or sodium could represent unique or special recipes that are still valuable in the analysis. Removing them could inadvertently remove interesting recipes that may contain common ingredients.

Distribution of the rating column

```
df.describe()

                fat      calories       protein        rating
sodium  \
count  1.590100e+04  1.596900e+04  15922.000000  20100.000000
1.596700e+04
mean   3.462407e+02  6.310443e+03     99.982665      3.713060
6.213949e+03
std    2.043552e+04  3.586637e+05   3836.459371      1.343144
3.329632e+05
min    0.000000e+00  0.000000e+00      0.000000      0.000000
0.000000e+00
25%    7.000000e+00  1.980000e+02      3.000000      3.750000
```

```
8.000000e+01
50%    1.700000e+01  3.310000e+02       8.000000       4.375000
2.940000e+02
75%    3.300000e+01  5.860000e+02      27.000000       4.375000
7.110000e+02
max    1.722763e+06  3.011122e+07  236489.000000       5.000000
2.767511e+07

       prep_time_minutes
count       17557.000000
mean          105.691918
std           183.321375
min             1.000000
25%            18.000000
50%            50.000000
75%           126.000000
max          4320.000000
```

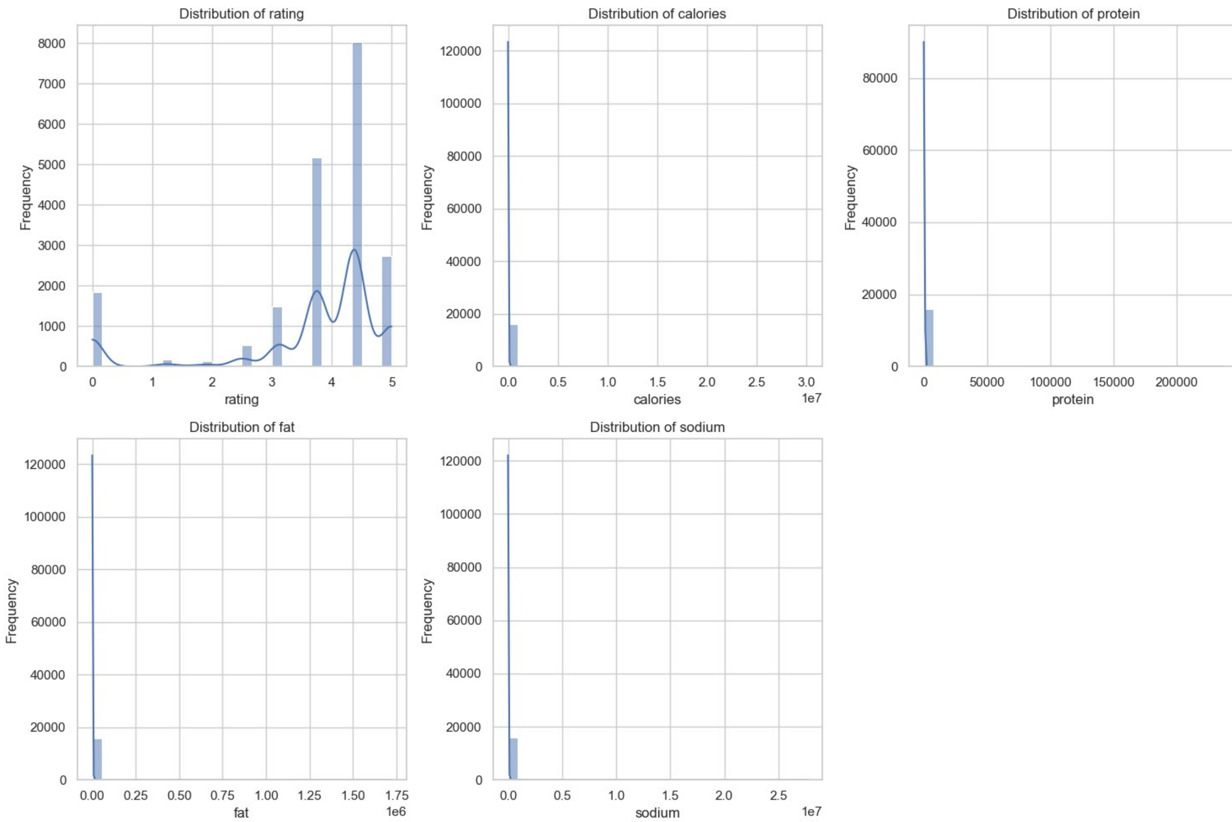
```
sns.set(style="whitegrid")

# Create a figure and axis
plt.figure(figsize=(10, 6))

# Plot the histogram for the 'ratings' column
sns.histplot(df['rating'], bins=30, kde=True)

# Add labels and title
plt.title('Distribution of Ratings', fontsize=16)
plt.xlabel('Ratings', fontsize=14)
plt.ylabel('Frequency', fontsize=14)

# Show the plot
plt.show()
```



The rating column ranges from 0 to 5, with a mean of 3.71, which suggests that most recipes are rated relatively positively.

distribution of other numeric columns

```
sns.set(style="whitegrid")
numerical_columns = ['rating', 'calories', 'protein', 'fat', 'sodium']

plt.figure(figsize=(15, 10))

for i, column in enumerate(numerical_columns):
    plt.subplot(2, 3, i + 1)
    sns.histplot(df[column].dropna(), bins=30, kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



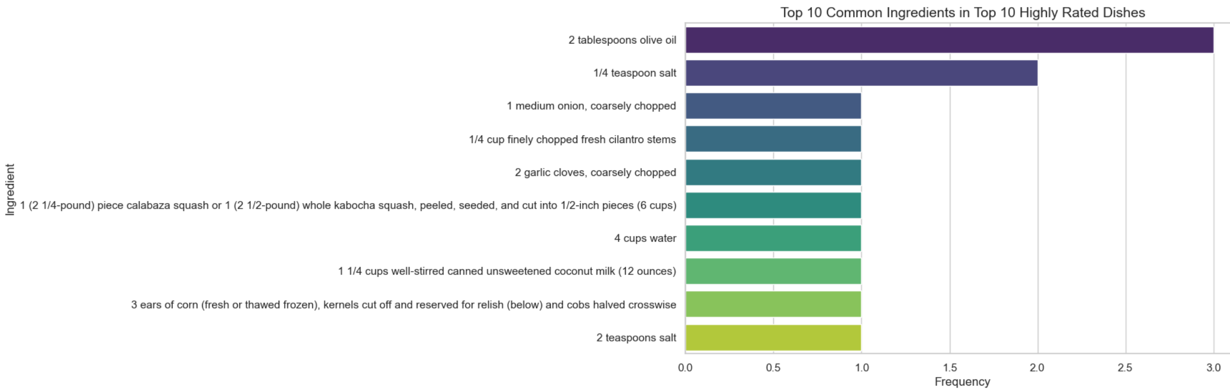
PROBLEM STATEMENT 1 = FIND OUT THE INGREDIENTS IN HIGHLY RATED RECIPIES

```python
top_10_dishes = df.sort_values(by='rating', ascending=False).head(10)

import ast
from collections import Counter

# Function to clean and flatten the ingredient list
def extract_ingredients(ingredient_str):
    try:
        # Convert string representation of list to actual list
        ingredients = ast.literal_eval(ingredient_str)
        return [ingredient.strip().lower() for ingredient in
ingredients]
    except:
        return []

# Extract ingredients from top 10 highly-rated dishes
top_10_ingredients =
top_10_dishes['ingredients'].apply(extract_ingredients)

# Flatten the list of ingredients across all 10 dishes
```

```python
flat_ingredients = [ingredient for sublist in top_10_ingredients for
ingredient in sublist]


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

# Get the 10 most common ingredients
top_10_common_ingredients = ingredient_counts.most_common(10)
print(top_10_common_ingredients)
```

```
[('2 tablespoons olive oil', 3), ('1/4 teaspoon salt', 2), ('1 medium
onion, coarsely chopped', 1), ('1/4 cup finely chopped fresh cilantro
stems', 1), ('2 garlic cloves, coarsely chopped', 1), ('1 (2 1/4-
pound) piece calabaza squash or 1 (2 1/2-pound) whole kabocha squash,
peeled, seeded, and cut into 1/2-inch pieces (6 cups)', 1), ('4 cups
water', 1), ('1 1/4 cups well-stirred canned unsweetened coconut milk
(12 ounces)', 1), ('3 ears of corn (fresh or thawed frozen), kernels
cut off and reserved for relish (below) and cobs halved crosswise',
1), ('2 teaspoons salt', 1)]
```

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

# Assuming top_10_common_ingredients is a list of tuples with
(ingredient, count)
# Convert the list of tuples into a DataFrame for easier plotting
common_ingredients_df = pd.DataFrame(top_10_common_ingredients,
columns=['Ingredient', 'Count'])

# Create a bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x='Count', y='Ingredient', data=common_ingredients_df,
palette='viridis')

# Add titles and labels
plt.title('Top 10 Common Ingredients in Top 10 Highly Rated Dishes',
fontsize=14)
plt.xlabel('Frequency', fontsize=12)
plt.ylabel('Ingredient', fontsize=12)

# Display the plot
plt.show()
```



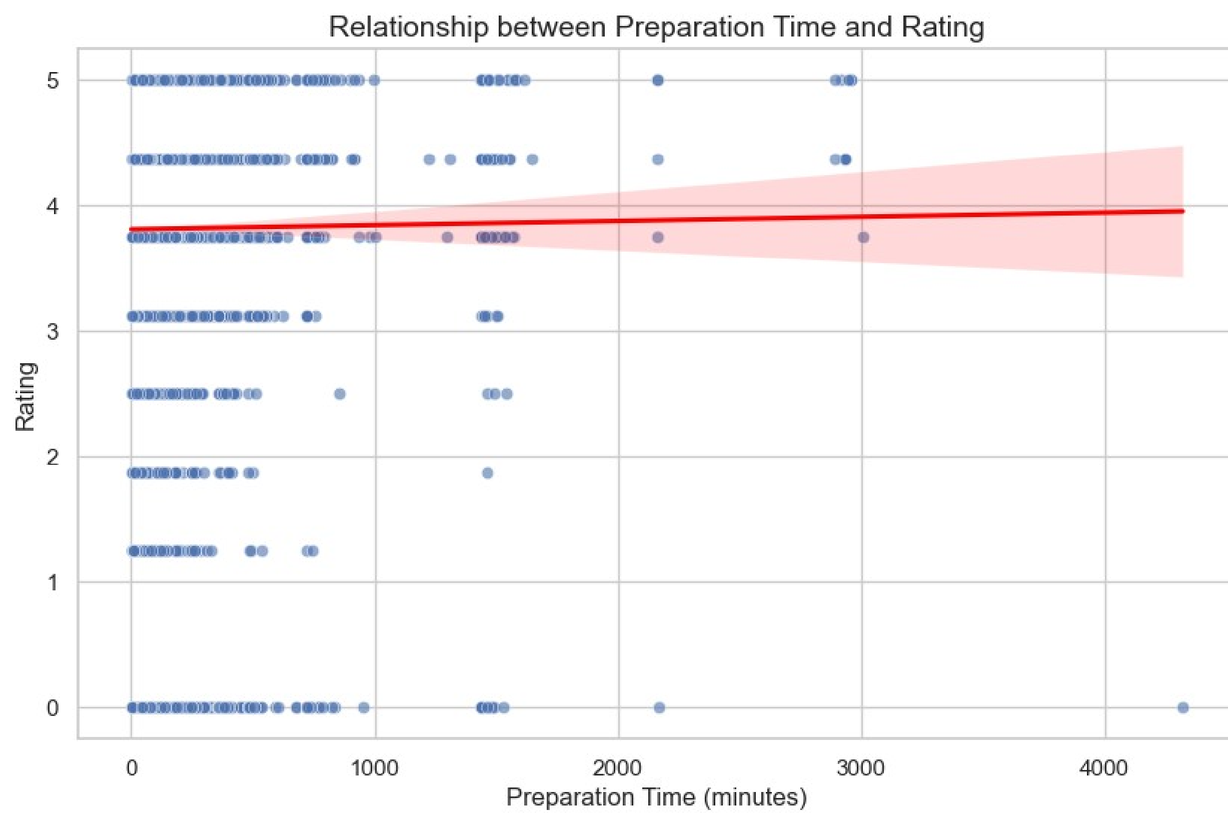
The above graph shows the 10 common ingredients in top rated dishes

PROBLEM STATEMENT 2 = WHAT IS THE CORELATION BETWEEN PREPERATION TIME AND RATINGS

There is no column which indicates the prep time , but we do have the directions column , from which we can extract times using regular expression analyse the relationship between prep time and ratings

```
import re

def extract_time(text):
    if pd.isna(text):
        return None

    hours = re.findall(r'(\d+)\s*hour', text)
    minutes = re.findall(r'(\d+)\s*minute', text)

    hours = int(hours[0]) if hours else 0
    minutes = sum(int(minute) for minute in minutes) if minutes else 0

    return hours * 60 + minutes if (hours or minutes) else None

# Apply the function to the 'directions' column
df['prep_time_minutes'] = df['directions'].apply(extract_time)

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.scatterplot(x='prep_time_minutes', y='rating', data=df, alpha=0.6)

sns.regplot(x='prep_time_minutes', y='rating', data=df, scatter=False,
```

```
color='red')

plt.title('Relationship between Preparation Time and Rating',
fontsize=14)
plt.xlabel('Preparation Time (minutes)', fontsize=12)
plt.ylabel('Rating', fontsize=12)

plt.show()
```



We can see from the above graph that the prep time has less or no effect on the ratings

PROBLEM STATEMENT 3 = HOW CAN THE DATA HELP IMPROVE USER EXPERIENCE FOR A RECIPE PLATFORM

1. Personalized Recipe Recommendations:

Top Ingredients: Use the most common ingredients found in highly-rated recipes to suggest personalized recommendations. For example, if a user frequently cooks with ingredients like garlic or olive oil, the platform can recommend recipes containing those ingredients, especially if they're found in high-rated dishes. Popular Recipes: Highlight recipes with the highest ratings that feature those top ingredients, increasing the chance of user satisfaction.

1. Improved Recipe Filtering:

Ingredient-Based Filtering: Allow users to search for recipes by specific ingredients. If a user enters an ingredient like "chicken" or "garlic," the platform can prioritize highly-rated recipes containing these ingredients. Cooking Time Filtering: Based on the analysis of preparation times, the platform can allow users to filter recipes by cooking time (e.g., "quick meals" under 30 minutes), which would cater to users seeking convenience.

1. Data-Driven Recipe Development:

Popular Ingredient-Based New Recipes: Chefs or content creators on the platform could develop new recipes using the most common ingredients from highly-rated dishes. This increases the likelihood of user acceptance, since these ingredients are proven to be popular. Recipe Optimization: The platform can optimize poorly rated recipes by identifying missing or underused top ingredients from the analysis, leading to more appealing and flavorful dishes.

1. User Engagement and Retention:

Personalized Meal Plans: Users can receive meal plans or recommendations based on their favorite ingredients or the most popular combinations seen in highly-rated recipes.

1. Better Health & Nutritional Insights:

Health-Conscious Suggestions: The analysis of nutritional values (e.g., calories, proteins, fats) allows the platform to recommend recipes based on dietary preferences, such as low-calorie or high-protein recipes, for health-conscious users. Balanced Recipes: Combining ingredients from high-rated recipes with nutritional insights, the platform can create and recommend more balanced or healthier meals.

1. Enhanced Search and Discovery:

Recipe Discovery Based on Ratings: Implement a feature that allows users to discover highly-rated recipes quickly, and show recipes that contain popular ingredients. This gives users confidence that they are choosing recipes others love. Dynamic Filters: Users could apply filters such as "top-rated," "quick-prep," or "common ingredients," based on the trends revealed in the dataset, improving search efficiency.

1. Optimize Recipe Instructions:

Preparation Time Guidance: From the analysis of prep time in the directions column, provide clear preparation time estimates. Recipes can be tagged with accurate time categories like "quick meals" or "long prep," improving user expectations and satisfaction.