Project Documentation Of Coffee Quality Dataset

TITLE : EDA of Coffee Quality Dataset
NAME : Shreeman DR
DA/DS : Data Analytics
Batch No. : B1(E) DA AltEd
Online/Offline : Online
Roll Number : 150324OLRE02

MILESTONE - I

Table of Contents

NOTE : After Project Workflow the Codes and other Contents will be displayed with Codes and Visualizations

1) Introduction

The Coffee Quality Database presents a rich repository of information encompassing various aspects of coffee production and quality evaluation. The objective of this project is to conduct a comprehensive data exploration, cleaning, and analysis to derive meaningful insights from the dataset. This Dataset contains the following features.

Quality Measures

- Aroma
- Flavour
- Aftertaste
- Acidity
- Body
- Balance
- Uniformity
- Cup Cleanliness
- Sweetness
- Moisture
- Defects

Bean Metadata

- Processing Method
- Colour
- Species

Farm Metadata

- Owner
- Country of Origin
- Farm Name
- Lot Number Mill
- Company
- Altitude
- Region

2) Aim

The aim of this project is to comprehensively explore, clean, and analyze the Coffee Quality Database to derive meaningful insights into the factors influencing coffee quality.
By examining various quality measures, bean metadata, and farm metadata, the project seeks to identify key trends, patterns, and anomalies that can inform improvements in coffee production processes, enhance quality assessment, and support decision-making in coffee sourcing and production optimization.

3) Problem Statement

Despite the rich information available in the Coffee Quality Database, many stakeholders in the coffee industry struggle to leverage this data effectively to improve coffee quality and production processes. The primary business problem is the lack of actionable insights derived from this data, which could enhance the quality assessment, optimize production methods, and inform sourcing decisions.
This project aims to bridge this gap by performing a thorough data analysis to uncover patterns, trends, and relationships that can guide better decision-making and ultimately improve the overall quality and consistency of coffee products.

- 👉 **What factors most significantly influence the quality of coffee as measured by attributes such as aroma, flavor, and aftertaste?**
- 👉 **How do different processing methods affect the overall quality and specific attributes of the coffee beans?**
- 👉 **What are the relationships between farm metadata (e.g., country of origin, altitude, and region) and coffee quality measures?**
- 👉 **Are there any noticeable patterns or trends in coffee quality based on the species of coffee beans (arabica vs. robusta)?**
- 👉 **How do specific farm-level practices and attributes (e.g., mill, lot number, company) correlate with coffee quality outcomes?**
- 👉 **What inconsistencies or anomalies exist in the data, and how might they impact the accuracy and reliability of quality assessments?**
- 👉 **What insights can be derived from visualizing the distribution and relationships within the dataset that could inform better production and sourcing decisions?**
- 👉 **What recommendations can be made to improve coffee quality based on the findings from the data analysis?**

These questions help to focus the analysis on extracting actionable insights that address the core issues faced by stakeholders in the coffee industry.

4) Project Overflow

Overview of the project workflow or methodology followed.

➡ Data Cleaning

Handling Missing Value Imputation and Removal, Inconvenience, Outlier Detection and Shape Comparison before and after.

➡ Exploratory Data Analysis (EDA)

Univariate Analysis, Segmented Univariate Analysis, Bivariate Analysis and Multivariate Analysis

➡ Data Visualization

- Utilize appropriate visualization techniques (histograms, box plots, scatter plots, etc.) to visually represent the distribution and relationships within the dataset.
- Create insightful visualizations to highlight patterns, trends, and anomalies in the data, facilitating better understanding and interpretation.

➡ Analysis and Interpretation

- Interpret findings from the data exploration and visualization.
- Identify key trends, patterns, or outliers that may impact the features.
- Offer recommendations or suggestions based on the analysis

➡ Documentation.

- Document the entire data exploration and analysis process, including methodologies, findings, and interpretations.
- Present the analysis report in a clear and concise manner, catering to both technical and non-technical audiences.

5) Data Understanding:

- **Description of the dataset, including structure, dimensions, and data types.**
- **Summary statistics and insights gained from initial data exploration.**

The dataset includes various measures of coffee quality such as aroma, flavor, aftertaste, acidity, body, balance, uniformity, cup cleanliness, sweetness, moisture, and defects. It also includes metadata about the beans (processing method, color, species) and the farms (owner, country of origin, farm name, lot number, mill, company, altitude, region).

Summary statistics and initial exploration provide insights into the distribution and characteristics of these variables, helping to identify potential areas of interest and concern.

Insights gained from initial data exploration.

- There are 1339 rows and 44 columns in the Dataset.
- From the info we conclude that Quality Measures Columns are all have Numerical Values, Bean Metadata are Categorical Values.
- Farm Metadata has both Categoriacal & Numerical Values.
- From the above information all mean of Quality Measures are close to each other - between 7.40 to 9.85 except Quakers(Defects) which is 0.17.
- All standard deviation of Quality Measures are close to each other - between 0.370064 to 0.832121 except Aroma which is 5.53.
- All Quality Measure Columns min-value starts from 0.00.

6) Data Cleaning

Techniques and methods used for cleaning the data, including handling missing values, outliers, and inconsistent entries.

➤➤ Handle missing values :

- **Identify Missing Values :** Used Heatmap and isnull().sum() to identify the missing values in each column
- **Droping the Columns :** Check the percentage of missing values in each column and remove which has more than 30% of missing values and unnecessary columns.
- **Imputating Missing Values :** Replacing missing values with Median & Mode according to the data type. After Replacing Check with Heatmap and isna().sum().

➤➤ Addressing Inconsistencies :

- Standardizing and Converting "Bag.Weight" Column into column for missing value replacement.
- Standardizing the "Harvest.Year" column for missing value replacement and further analysis.
- Standardizing the "Altitude" column for missing value replacement and further analysis.

➤➤ Handling Outliers :

- **Identifying of Outliers :**
  Used Boxplot and Scatterplot to visually identify the outliers. Z-Score for accurate index of Outliers.
- **Removal of Outliers :**
  Used df2.drop( ) to remove all the outliers index in the dataset.
- **Shape Comparison/Checking :**
  Comparing the shape of the dataset before and after outlier removal.

NOTE : All codes for Data Cleaning are given below 👇 in the pdf.

7) Obtaining Derived Metrics :

➢ *Description of derived metrics created to enhance the dataset.*

- ⚡ Created a Total Quality Score from Quality Measures like 'Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body' and 'Balance'.
- ⚡ Created a Column with Aroma to Flavor Ratio in df2.
- ⚡ Made a Normalised Column out of Quality Measures Columns such as Normalized Aroma , Flavor, Aftertaste, Acidity, Body, and Balance.

Finally displaying Summary statistics of the new derived metrics

NOTE : All codes of derived Matrix is given below 👇 in the pdf.

8) Filtering Data for Analysis :

Explanation of any additional data filtering or preprocessing steps performed to prepare the data for analysis.

✅ First Assigned columns which belongs to Quality_Measures, Bean_Metadata and Farm Metadata.

Further Classifying the Columns based on the Type of Data it has

This filtering will help during the Chi-Square. lets check which columns are Categorical Data in Non-Numeric Columns. Iam going to further classify the Non_Numerical_Data into Categorical_Data, Unique_Data, and Date_Data.

✨ The Columns which have less than 100 unique values are conisdered as Categorical_Data.
✨ The Columns which has more than 100 unique values are conisdered as Unique_Data.
✨ The Columns which have date values in it are Date_Data.

```
Numericals_Data = df2[['Number.of.Bags', 'Bag.Weight', 'Altitude','Aroma', 'Flavor', 'Aftertaste','Acidity', 'Body',
                       'Balance', 'Uniformity', 'Cupper.Points', 'Clean.Cup', 'Sweetness', 'Moisture', 'Quakers',
                       'Defects']]
Categorical_Data = df2[['Country.of.Origin', 'In.Country.Partner', 'Variety', 'Species', 'Processing.Method',
Date_Data.       = df2[['Grading.Date', 'Expiration', 'Harvest.Year']]
Unique_Data      = df2[['Owner', 'Farm.Name', 'Mill', 'Company', 'Region', 'Producer']]
```

NOTE: All codes of Filtering Data for Analysis is given below 👇 in the pdf

EXPLORATORY DATA ANALYSIS

9) EDA - Univariate Analysis :

➢ *Insights gained from univariate analysis, including visualizations of individual variables.*

- ➡ Visualisation of Distribution and Frequency of Species, Color and Processing.Method of Coffee by Bar Chart and Pie Chart.
- ➡ Visualisation of Count Distributions of Quality Measures of Coffee by Histogram.
- ➡Checking the Symmetrical Distribution of all Numerical Columns in a Dataset.

Insights gained from Univariate Analysis :

- ◉ **Acidity :** There's a possible normal distribution (bell-shaped curve) indicating most responses fall around the average acidity level.
- ◉ **Body :** There's a possible right-skewed distribution suggesting more ratings towards a lighter body.
- ◉ **Balance and Uniformity :** The results are inconclusive due to missing labels on the axes.
- ◉ **Other Features :** There were mentions of factors like Cupper Points, Clean Cup score, sweetness, moisture, and Quakers (which likely meant something else taste-related). However, the exact details and interpretations were unclear due to missing labels or potential data entry errors.

Overall from the chart it suggests the coffee might be on the lighter side in terms of acidity and body. Here are some additional questions that could be answered with more information:

What roast profile was used for the coffee?
What brewing method was used?
What is the origin of the coffee beans?
What do the labels for the axes in the graphs represent?

Knowing these details would allow for a more comprehensive analysis of the coffee.

NOTE: All codes of EDA - Univariate Analysis is given below 👇 in the pdf

EXPLORATORY DATA ANALYSIS

10) Segmented Univariate Analysis :

Analysis of data segments or categories to gain deeper insights.

➡ Used Violin Plot and 3D Scatterplot for Segmented Univariate Analysis.

Insights gained from Segmented Univariate Analysis :

Insights from the Violin Plot :-

The violin plot illustrates the distribution of the "Aroma" scores across different "Processing Methods". The key observations are:

Washed / Wet Method:

- This method shows the highest median Aroma score among the methods displayed.
- The distribution is wide, indicating a high variability in Aroma scores.

Natural / Dry Method:

- The median Aroma score for this method is lower than the Washed/Wet method.
- It also shows a wide distribution, similar to the Washed/Wet method, indicating considerable variability.

Pulped natural / Semi-washed Method:

- This method shows a moderate median Aroma score.
- The distribution is more compact compared to the Washed/Wet and Natural/Dry methods, suggesting less variability in Aroma scores.

Semi-pulped Method:

- This method shows a lower median Aroma score.
- The distribution is relatively narrow, indicating consistent Aroma scores within this method.

Other Methods:

- These methods show the lowest median Aroma scores.
- The distribution is quite narrow, indicating less variability in Aroma scores.

Insights from the 3D-Scatterplot :-

A 3D scatterplot shows the relationship between three variables simultaneously, revealing patterns, clusters, and potential correlations in three-dimensional space.

- The Aroma Flavor and Acidity have closely clustered and not much spread around which means they most of the points are on the same high scoring to Total Coffee Points.
- Most of the Values are around 7 - 10.

NOTE: All codes of Segmented Univariate Analysis is given below 👇 in the pdf.

EXPLORATORY DATA ANALYSIS

11) Bivariate Analysis :

Analysis of relationships between pairs of variables.

➡ Used Correlation Matrix, Chi2 Test and ANOVA Fclassif Test.

Insights from the Correlations :-

◉ Positive & Negative Correlation :
The data points form a clear upward trend, indicating a strong positive correlation between "Aftertaste" and "Flavor" whereas the Aftertaste vs Category.One.Defects is negatively correlated.

◉ Proportional Effect :
This means that as the aftertaste score increases, the flavor score also tends to increase whereas the opposite effect on Aftertaste and Category.One.Defects.

◉ Clustered Points :
Most data points are clustered between aftertaste scores of around 6 to 8 and flavor scores of around 6 to 8. This suggests that the majority of the samples have high scores in both aftertaste and flavor.

Insights from the Chi2 Test :-

🔎 Significant Associations:

└➤ Species and Country of Origin: The chi-squared statistic is 799.09 with a very low p-value (3.099531e-145), indicating a very strong association between coffee species and country of origin.

└➤ In Country Partner and Country of Origin : The chi-squared statistic is 15106.81 with a p-value close to zero, indicating a strong association between in-country partner and country of origin.

└➤ Variety and Country of Origin : The chi-squared statistic is 7651.53 with a p-value of zero, indicating a strong association between coffee variety and country of origin.

└➤ Processing Method and Country of Origin : The chi-squared statistic is 793.85 with a p-value (6.006653e-92), showing a significant association between processing method and country of origin.

└➤ Color and Country of Origin : The chi-squared statistic is 226.85 with a p-value (1.918919e-18), indicating a significant association between coffee color and country of origin.

🔎 Moderate Associations :

└➤ Species and In Country Partner : The chi-squared statistic is 178.88 with a p-value (3.328648e-25), suggesting a significant, though not as strong, association between species and in-country partner.

└➤ Processing Method and In Country Partner : The chi-squared statistic is 638.60 with a p-value (2.108365e-79), indicating a significant association.

└➤ Variety and In Country Partner : The chi-squared statistic is 5396.60 with a p-value of zero, showing a strong association.

🔎 Weak or No Associations:

└➤ Processing Method and Species : The chi-squared statistic is 2.23 with a high p-value (6.931748e-01), suggesting no significant association.

└➤ Color and Species : The chi-squared statistic is 0.99 with a p-value (6.093336e-01), indicating no significant association.

└➤ Color and Processing Method : The chi-squared statistic is 44.42 with a p-value (4.743247e-07), which is significant but not as strong compared to other associations.

🚀 Analysed Relationships/ Intrepretation :

💥 Strongest Associations : The Country of Origin appears to be a key factor influencing various other attributes of coffee, such as species, in-country partner, variety, processing method, and color.

💥 Moderate Associations : Attributes like species and in-country partner have notable associations, which might be relevant for understanding regional coffee characteristics.

💥 Weak Associations : There is little to no significant relationship between processing method and species, as well as color and species, suggesting these attributes may vary independently of each other.

NOTE: All codes of Bivariate Analysis is given below 👇 in the pdf.

EXPLORATORY DATA ANALYSIS

11) Bivariate Analysis :

Analysis of relationships between pairs of variables.

➡ Used Correlation Matrix, Chi2 Test and ANOVA Fclassif Test.

Insights from the ANOVA Test :-

Selected Features for Coffee Quality Measures

No. of Bags & Sweetness :

➤ Number.of.Bags: Highly significant (F Stat: 25.480386, P-value: 1.764103e-121) influencing quality measures.

➤ Sweetness: Very significant (F Stat: 16.249817, P-value: 8.014149e-79) affecting various quality attributes.

Top ANOVA Results for Coffee Quality

➤ Balance: Highly significant (F Stat: 11.983125, P-value: 5.317317e-57).
➤ Aftertaste: Strong impact (F Stat: 11.667931, P-value: 2.461880e-55).
➤ Body: Significant (F Stat: 11.667481, P-value: 2.475422e-55).
➤ Sweetness: Notable influence (F Stat: 16.249817, P-value: 8.014149e-79).

Summary

Key Influencers:
Number of Bags, Sweetness, Balance, Aftertaste, and Body are major factors affecting coffee quality.

Processing Method:
Body and Quakers are crucial for determining the processing method's impact.

These insights can guide stakeholders in the coffee industry, such as farmers, processors, and marketers, to focus on these critical factors to enhance coffee quality.

Understanding these relationships can help in making informed decisions about cultivation practices, processing methods, and partnership selections to achieve desired quality outcomes.

NOTE: All codes of Bivariate Analysis is given below 👇 in the pdf.

EXPLORATORY DATA ANALYSIS

12) Multivariate Analysis

Exploration of complex relationships involving multiple variables.

➡ Used Clustering, lmplot, and Pairplot.

Results from Clustering: Aroma vs Flavor

- ➤ Most of points in KMean Cluster plot of x-axis as "Aroma" and y-axis as "Flavor".
- ➤ Most of the data Points of both were between 6 - 10 .
- ➤ The observed patterns in the clusters indicate the diversity of aroma and flavor profiles within the dataset.
- ➤ The clustering can also highlight any outliers or unique combinations of these features, which might be worth further exploration to understand their uniqueness or potential market value.

Results from lmplot

Aftertaste vs Flavor by Species :

- ➤ The data points of both species of coffee beans have been close to ranging from 6-10.
- ➤ Even though 'Robusta' is smaller than 'Arabica' in distribution the scores of these attributes are higher focus on 6.5-8.
- ➤ There is a linear line between them.

Aroma vs Body by Processing.Method :

- ➤ By this lmplot all the the data points are are very close to each other. This means the scores are similar to all methods ranging between 6 to 9/10.
- ➤ Out of all the processing methods the most used is Washed/Wet Method

Results from Pairplot :

- ➤ Comparison with all the attributes with hue as Processing Methods.

NOTE: All codes of Multivariate Analysis is given below 👇 in the pdf.

13) Overall Insights obtained from Analysis :

✨ *Summary of key insights and findings obtained from the analysis.*

Key Factors :

🚀 Key Factors Affecting the Quatity of Coffee :

By doing EDA, I conclude that Quality measures 'Flavor', 'Aftertaste', 'Balance', 'Body', 'Moisture' and 'Defects' affects the Quality of the Coffee.

- ✳ Quality Measures like Flavor, Aftertaste, Balance and Body have a positive effect
- ✳ Whereарea Moisture and Defects (Category.One.Defects & Category.One.Defects)

🚀 Coffee Bean Factors on Quality of Coffee :

Among the factors that affect coffee quality, several stand out as particularly influential are Species, Processing Method, Country of Origin, In-Country Partner and Variety.

🚀 Understanding the Relationships between Variables :

- 💥 **The Bean_Metadata and Farm_Metadata have a signification relationship between The Total Quality Score of the Coffee.**
  - **Processing.Method, In.Country.Partner and Country.of.Origin.**
- 💥 **Specific farm attributes, such as altitude, have a impacts on coffee processing method (dry/roasting).**
  - **Altitude have a inner relationship Moisture which gives impact.**
  - **The Higher the Moisture is the method will change according to Washed/Wet and Dry/Roast.**
- 💥 **Certain processing methods and regions consistently produce higher quality coffee.**
  - **The regions using the Washed/ Wet processing method is the most and have a goodscore on the Coffee Quality.**

Important Variables :

- ✳ Flavor, Aftertaste, Acidity, Body and Balance. Total.Cup.Points which I have used to find.
- ✳ Processing Methods, Variety, Country Of Origin and Altitude.

Dependencies between Columns :

- ✳ Flavor and Aftertaste are strongly correlated, including that improvements in the Aroma may lead to enhancements in Flavor.
- ✳ Species and Country.of.Origin have a great relationship between them.
- ✳ Country.of.Origin and Processing.Method are also strongly related which means different countries use different methods to process the coffee.
- ✳ which results directly on Processing.Method and Flavor & Aftertaste of the coffee.

INTERESTING FACT 🤔

Processing.Method is the most important factor which impacts the Qualities of the Coffee except for the Quakers and Defects and Defect.

13) Overall Insights obtained from Analysis :

✨ *Summary of key insights and findings obtained from the analysis.*

Recommendations for Improving Coffee Quality and Production Practices :

Based on the analysis conducted on the Coffee Quality Dataset, here are several recommendations for the clients aimed at improving coffee quality and production processes: Recommendations for Improving Coffee Quality

1. Focus on Key Quality Attributes:

The analysis indicates that attributes such as Flavor, Aftertaste, Balance, and Body are strongly correlated with overall coffee quality. Emphasize improving these aspects through targeted cultivation and processing techniques.

2. Optimize Processing Methods:

Different processing methods significantly impact coffee quality. Invest in training and infrastructure to adopt the processing methods that have shown to yield higher quality coffee. Experiment with various methods to find the optimal ones for specific coffee bean varieties.

3. Leverage Regional Strengths:

Regions and farms with consistently high-quality coffee should be studied and emulated. Encourage knowledge sharing and best practices among farmers in different regions. Promote regional coffee as a premium product if it consistently meets high-quality standards.

4. Reducing Defects:

I recommend implementing the farming practices on reducing the Defects , Quakers and maintaining the moisture levels because it can lower the quality of the Coffee.

5. Market Segmentation:

Divided by the Overall Coffee Score on the Derived Metrix, I would suggest that segment the market based on quality and cater to different customer segments with appropriate pricing strategies. High-quality coffee can be marketed as a premium product, while other segments can focus on cost-effective production for mass markets.

Strategic Recommendations

1. Research and Development:

Invest in R&D to explore new varieties of coffee beans that might offer better quality or resilience against diseases and climate change. Collaborative research with agricultural universities or institutions can yield valuable insights.

2. Consumer Education:

Educate consumers about the different aspects of coffee quality and the efforts made to ensure high standards. This can help in building a loyal customer base willing to pay a premium for quality coffee.

3. Technology Adoption:

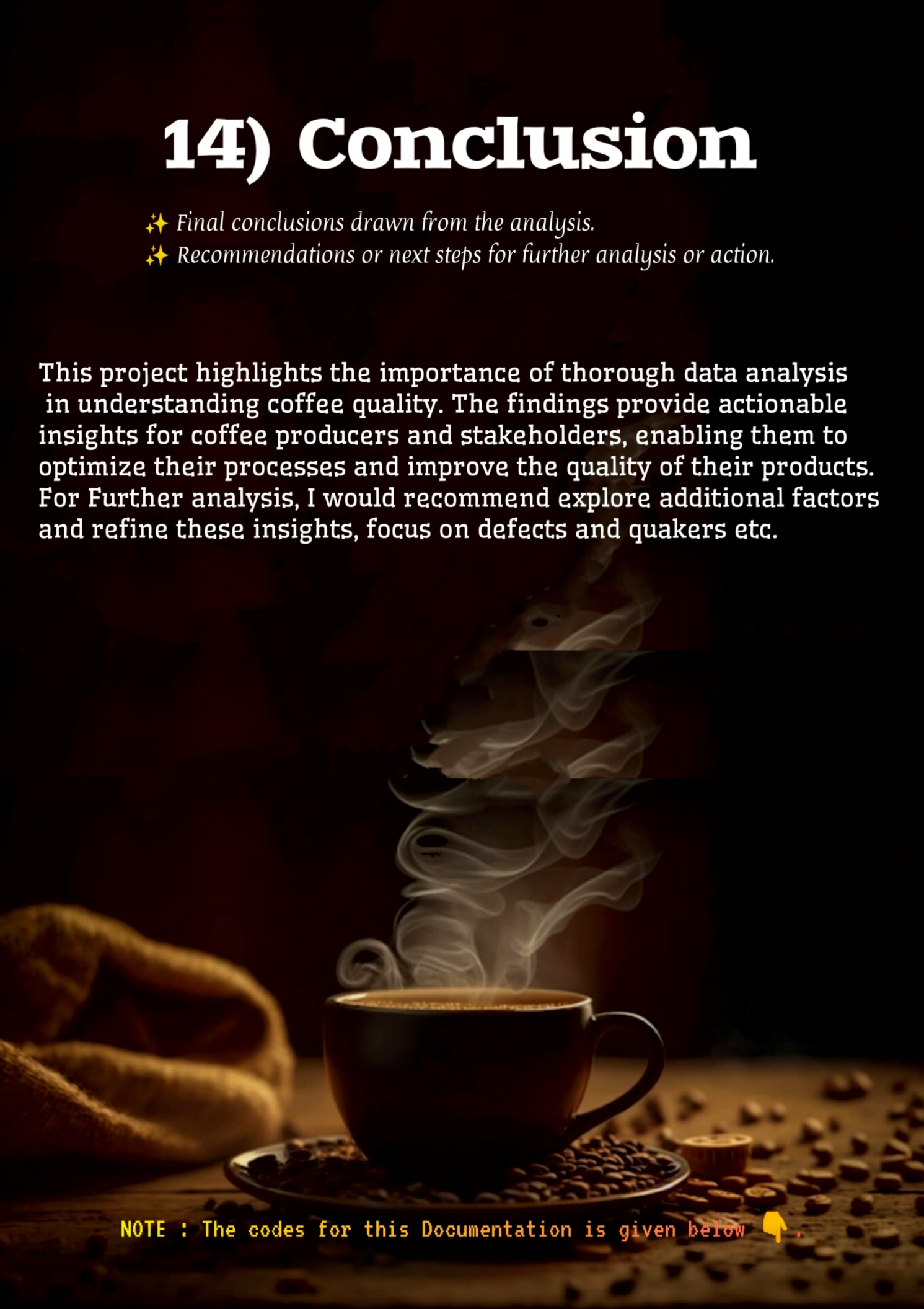
Encourage the adoption of technology in various stages of coffee production. Precision agriculture, automation in processing, and advanced analytics can lead to more efficient and higher-quality production.

By focusing on these recommendations, clients can enhance the quality of their coffee, optimize production processes, and potentially increase their profitability. Continuous improvement and adaptation to new findings will be crucial in maintaining a competitive edge in the coffee industry.

14) Conclusion

✨ *Final conclusions drawn from the analysis.*
✨ *Recommendations or next steps for further analysis or action.*

This project highlights the importance of thorough data analysis in understanding coffee quality. The findings provide actionable insights for coffee producers and stakeholders, enabling them to optimize their processes and improve the quality of their products. For Further analysis, I would recommend explore additional factors and refine these insights, focus on defects and quakers etc.



NOTE : The codes for this Documentation is given below 👇.

Milestone-1 Evaluation Project

Project Documentation: Exploratory Data Analysis of Coffee Quality Dataset :

Title : Coffee Quality Data Analysis

Name : Shreeman D.R.

DA/DS : Data Analytics (DA)

Batch number : B1(E) DA AltEd

Online/Offline : Online

Roll Number : 150324OLRE02

Table of Contents :

NOTE : All the codes used for this are given after the documentation and displaying of results.

==========

1) Introduction :

The Coffee Quality Database presents a rich repository of information encompassing various aspects of coffee production and quality evaluation.
The objective of this project is to conduct a comprehensive data exploration, cleaning, and analysis to derive meaningful insights from the dataset..

Quality Measures:

- **Aroma**
- **Flavour**
- **Aftertaste**
- **Acidity**
- **Body**
- **Balance**
- **Uniformity**
- **Cup Cleanliness**
- **Sweetness**
- **Moisture**
- **Defects**

Bean Metadata:

- **Processing Method**
- **Colour**
- **Species**

Farm Metadata:

- **Owner**
- **Country of Origin**
- **Farm Name**
- **Lot Number**
- **Mill**
- **Company**
- **Altitude**
- **Region**

2) Aim :

The aim of this project is to comprehensively explore, clean, and analyze the Coffee Quality Database to derive meaningful insights into the factors influencing coffee quality.
By examining various quality measures, bean metadata, and farm metadata, the project seeks to identify key trends, patterns, and anomalies that can inform improvements in coffee production processes, enhance quality assessment, and support decision-making in coffee sourcing and production optimization.

==========

3) Problem Statement :

Despite the rich information available in the Coffee Quality Database, many stakeholders in the coffee industry struggle to leverage this data effectively to improve coffee quality and production processes. The primary business problem is the lack of actionable insights derived from this data, which could enhance the quality assessment, optimize production methods, and inform sourcing decisions. This project aims to bridge this gap by performing a thorough data analysis to uncover patterns, trends, and relationships that can guide better decision-making and ultimately improve the overall quality and consistency of coffee products.

👉 What factors most significantly influence the quality of coffee as measured by attributes such as aroma, flavor, and aftertaste?
👉 How do different processing methods affect the overall quality and specific attributes of the coffee beans?
👉 What are the relationships between farm metadata (e.g., country of origin, altitude, and region) and coffee quality measures?
👉 Are there any noticeable patterns or trends in coffee quality based on the species of coffee beans (arabica vs. robusta)?
👉 How do specific farm-level practices and attributes (e.g., mill, lot number, company) correlate with coffee quality outcomes?
👉 What inconsistencies or anomalies exist in the data, and how might they impact the accuracy and reliability of quality assessments?
👉 What insights can be derived from visualizing the distribution and relationships within the dataset that could inform better production and sourcing decisions?
👉 What recommendations can be made to improve coffee quality based on the findings from the data analysis?

These questions help to focus the analysis on extracting actionable insights that address the core issues faced by stakeholders in the coffee industry.

==========

4) Project Workflow :

Overview of the project workflow or methodology followed.

- Data Cleaning
- Exploratory Data Analysis (EDA)
- Data Visualization
- Analysis and Interpretation
- Documentation

==========

5) Data Understanding :

➢ *Description of the dataset, including structure, dimensions, and data types.*
➢ *Summary statistics and insights gained from initial data exploration.*

Insights gained from initial data exploration

- There are 1339 rows and 44 columns in the Dataset.
- From the info we conclude that Quality Measures Columns are all have Numerical Values, Bean Metadata are Categorical Values.
- Farm Metadata has both the Values - Categoriacal and Numerical Values.
- From the above information all mean of Quality Measures are close to each other - between 7.40 to 9.85 except Quakers(Defects) which is 0.17.
- All standard deviation of Quality Measures are close to each other - between 0.370064 to 0.832121 except Aroma which is 5.53.
- All Quality Measure Columns min-value starts from 0.00.

In [44]:
```
data.head()
```

Out [44]:

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Catego
0	0	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0
1	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1
2	2	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0
3	3	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2
4	4	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2

5 rows × 44 columns

In [48]:
```
data.info() #📣to get some basic info on the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1339 entries, 0 to 1338
Data columns (total 44 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             1339 non-null   int64
 1   Species                1339 non-null   object
```

```
 2   Owner                   1332 non-null   object
 3   Country.of.Origin       1338 non-null   object
 4   Farm.Name               980 non-null    object
 5   Lot.Number              276 non-null    object
 6   Mill                    1021 non-null   object
 7   ICO.Number              1180 non-null   object
 8   Company                 1130 non-null   object
 9   Altitude                1113 non-null   object
 10  Region                  1280 non-null   object
 11  Producer                1107 non-null   object
 12  Number.of.Bags          1338 non-null   float64
 13  Bag.Weight              1339 non-null   object
 14  In.Country.Partner      1339 non-null   object
 15  Harvest.Year            1292 non-null   object
 16  Grading.Date            1339 non-null   object
 17  Owner.1                 1332 non-null   object
 18  Variety                 1113 non-null   object
 19  Processing.Method       1169 non-null   object
 20  Aroma                   1339 non-null   float64
 21  Flavor                  1339 non-null   float64
 22  Aftertaste              1339 non-null   float64
 23  Acidity                 1339 non-null   float64
 24  Body                    1339 non-null   float64
 25  Balance                 1339 non-null   float64
 26  Uniformity              1339 non-null   float64
 27  Clean.Cup               1339 non-null   float64
 28  Sweetness               1339 non-null   float64
 29  Cupper.Points           1339 non-null   float64
 30  Total.Cup.Points        1339 non-null   float64
 31  Moisture                1339 non-null   float64
 32  Category.One.Defects    1339 non-null   int64
 33  Quakers                 1338 non-null   float64
 34  Color                   1069 non-null   object
 35  Category.Two.Defects    1339 non-null   int64
 36  Expiration              1339 non-null   object
 37  Certification.Body      1339 non-null   object
 38  Certification.Address   1339 non-null   object
 39  Certification.Contact   1339 non-null   object
 40  unit_of_measurement     1339 non-null   object
 41  altitude_low_meters     1109 non-null   float64
 42  altitude_high_meters    1109 non-null   float64
 43  altitude_mean_meters    1109 non-null   float64
dtypes: float64(17), int64(3), object(24)
memory usage: 460.4+ KB
```

In [50]:

```
#📣 Checking the shape of the Data :
print("Number of Rows:",data.shape[0])
print("Number of Columns:",data.shape[1])
```

```
Number of Rows: 1339
Number of Columns: 44
```

In [54]:

```
#📣 getting some basic stats of the Dataset:
data.describe()
display(Markdown(bold_text))
print('Stats for Numerical DataTypes')
data.describe()
```

```
<IPython.core.display.Markdown object>
```

```
Stats for Numerical DataTypes
```

Out [54]:

	Unnamed: 0	Number.of.Bags	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Clean.C
count	1339.000000	1338.000000	1339.000000	1339.000000	1339.000000	1339.000000	1339.000000	1339.000000	1339.000000	1339.0000
mean	669.000000	159.085202	7.770187	7.520426	7.401083	7.535706	7.517498	7.518013	9.834877	9.835108
std	386.680316	173.698167	5.534440	0.398442	0.404463	0.379827	0.370064	0.408943	0.554591	0.763946
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	334.500000	14.000000	7.420000	7.330000	7.250000	7.330000	7.330000	7.330000	10.000000	10.000000
50%	669.000000	175.000000	7.580000	7.580000	7.420000	7.580000	7.500000	7.500000	10.000000	10.000000
75%	1003.500000	275.000000	7.750000	7.750000	7.580000	7.750000	7.670000	7.750000	10.000000	10.000000
max	1338.000000	3200.000000	200.000000	8.830000	8.670000	8.750000	8.580000	8.750000	10.000000	10.000000

In [56]:

```
print('Stats for Non-Numerical DataTypes')
data.select_dtypes(include=['object', 'category']).describe()
```

```
Stats for Non-Numerical DataTypes
```

Out [56]:

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Grading.Date	Own
count	1339	1332	1338	980	276	1021	1180	1130	1113	1280	...	1339	1332
unique	2	315	36	571	227	459	845	281	396	356	...	567	319
top	Arabica	juan luis alvarado romero	Mexico	various	1	beneficio ixchel	0	unex guatemala, s.a.	1100	huila	...	July 11th, 2012	Juan Luis Alva Rom
freq	1311	155	236	47	18	90	79	86	43	112	...	25	155

4 rows × 24 columns

==========

6) Data Cleaning (Missing Values Imputation, Outliers, Handling Inconsistent Values):

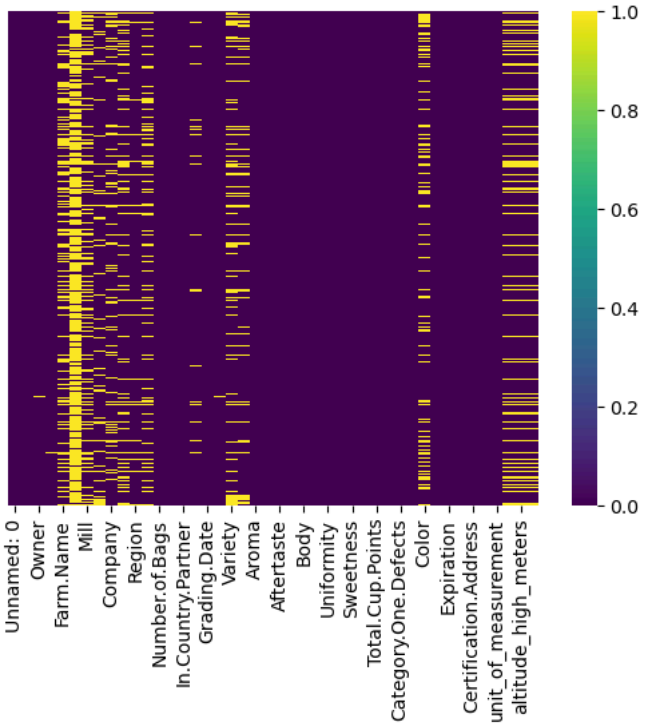
➢ *Techniques and methods used for cleaning the data, including handling missing values, outliers, and inconsistent entries.*

➽ Handle missing values :

└➢ Identify Missing Values

In [58]:
```
sns.heatmap(data.isnull(),yticklabels=False,cmap = 'viridis') #📣 checking the null values in the heatmap
```

Out [58]: <Axes: >



In [63]:
```
# 📣 Check for missing values
missing_values = data.isna().sum()
display(Markdown(bold_text1))
print(missing_values)
```

```
<IPython.core.display.Markdown object>

Unnamed: 0                0
Species                   0
Owner                     7
Country.of.Origin         1
Farm.Name               359
Lot.Number             1063
Mill                    318
ICO.Number              159
Company                 209
Altitude                226
Region                   59
Producer                232
Number.of.Bags            1
Bag.Weight                0
In.Country.Partner        0
Harvest.Year             47
Grading.Date              0
Owner.1                   7
Variety                 226
Processing.Method       170
Aroma                     0
Flavor                    0
Aftertaste                0
Acidity                   0
Body                      0
Balance                   0
Uniformity                0
Clean.Cup                 0
Sweetness                 0
Cupper.Points             0
Total.Cup.Points          0
Moisture                  0
Category.One.Defects      0
Quakers                   1
Color                   270
Category.Two.Defects      0
Expiration                0
Certification.Body        0
Certification.Address     0
Certification.Contact     0
unit_of_measurement       0
altitude_low_meters     230
altitude_high_meters    230
altitude_mean_meters    230
dtype: int64
```

Comment:
We can observe from the above heatmap and in numbers as well that the missing value in the "Lot.Number" column. It is not necessary for EDA. So we

can drop this column.
But Let's see the other columns as well. Let's drop the column which has the missing values is more then 30% of Total Values in the column.

In [75]:
```
#📣 Print the percentage of null values for each column
display(Markdown(bold_text2))
formatted_null_percentage
```

Out [75]:
```
<IPython.core.display.Markdown object>
Unnamed: 0                 0.00%
Species                    0.00%
Owner                      0.52%
Country.of.Origin          0.07%
Farm.Name                 26.81%
Lot.Number                79.39%
Mill                      23.75%
ICO.Number                11.87%
Company                   15.61%
Altitude                  16.88%
Region                     4.41%
Producer                  17.33%
Number.of.Bags             0.07%
Bag.Weight                 0.00%
In.Country.Partner         0.00%
Harvest.Year               3.51%
Grading.Date               0.00%
Owner.1                    0.52%
Variety                   16.88%
Processing.Method         12.70%
Aroma                      0.00%
Flavor                     0.00%
Aftertaste                 0.00%
Acidity                    0.00%
Body                       0.00%
Balance                    0.00%
Uniformity                 0.00%
Clean.Cup                  0.00%
Sweetness                  0.00%
Cupper.Points              0.00%
Total.Cup.Points           0.00%
Moisture                   0.00%
Category.One.Defects       0.00%
Quakers                    0.07%
Color                     20.16%
Category.Two.Defects       0.00%
Expiration                 0.00%
Certification.Body         0.00%
Certification.Address      0.00%
Certification.Contact      0.00%
unit_of_measurement        0.00%
altitude_low_meters       17.18%
altitude_high_meters      17.18%
altitude_mean_meters      17.18%
dtype: object
```

RESULTS

- The Columns which has more than 30% of null values will be removed = "Lot.Number".
- We also remove the columns because they contain null values and this column is not very useful for our analysis. = "Unnamed: 0", "ICO Number", "Certification.Address", "Certification.Contact", "unit_of_measurement", "altitude_low_meters", "altitude_high_meters" and "altitude_mean_meters".

Let's see the columns in the data (df) after removing those columns.

In [92]:
```
print('No. of Columns after removing -',a)
print('No. of Columns after removing -',df.shape[1])
df.columns
```

```
No. of Columns after removing - 44
No. of Columns after removing - 34
```

Out [92]:
```
Index(['Species', 'Owner', 'Country.of.Origin', 'Farm.Name', 'Mill', 'Company',
       'Altitude', 'Region', 'Producer', 'Number.of.Bags', 'Bag.Weight',
       'In.Country.Partner', 'Harvest.Year', 'Grading.Date', 'Owner.1',
       'Variety', 'Processing.Method', 'Aroma', 'Flavor', 'Aftertaste',
       'Acidity', 'Body', 'Balance', 'Uniformity', 'Clean.Cup', 'Sweetness',
       'Cupper.Points', 'Total.Cup.Points', 'Moisture', 'Category.One.Defects',
       'Quakers', 'Color', 'Category.Two.Defects', 'Expiration'],
      dtype='object')
```

Comment : As we can see here that those columns have been removed. Before it was 44 columns and now it has reduced to 34 columns.

↳➤ Replacing/Imputating Missing Values

In [244]:
```
# Replacement by median for Numerical Data USING Median and Non-Numerical Data USING Mode :
for column in All_Columns:
    if df[column].dtype == 'object':
        mode_value = df[column].mode()[0]
        print(f"Filling missing values in column '{column}' with mode: {mode_value}")
        df[column].fillna(mode_value, inplace=True)
    else:
        median_value = df[column].median()
        print(f"Filling missing values in column '{column}' with median: {median_value}")
        df[column].fillna(median_value, inplace=True)
```

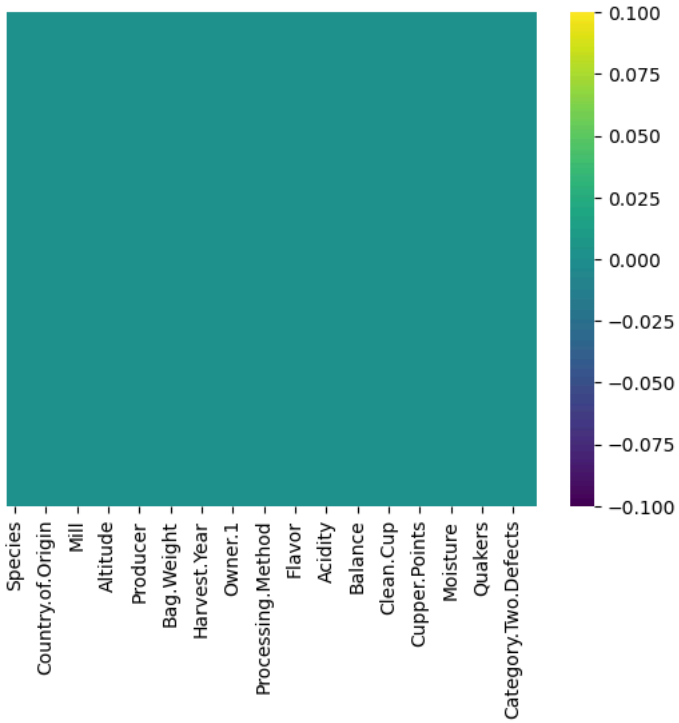
```
Filling missing values in column 'Species' with mode: Arabica
Filling missing values in column 'Owner' with mode: juan luis alvarado romero
Filling missing values in column 'Country.of.Origin' with mode: Mexico
Filling missing values in column 'Farm.Name' with mode: various
Filling missing values in column 'Mill' with mode: beneficio ixchel
Filling missing values in column 'Company' with mode: unex guatemala, s.a.
```

```
Filling missing values in column 'Altitude' with median: 1300.0
Filling missing values in column 'Region' with mode: huila
Filling missing values in column 'Producer' with mode: La Plata
Filling missing values in column 'Number.of.Bags' with median: 175.0
Filling missing values in column 'Bag.Weight' with median: 45.3592
Filling missing values in column 'In.Country.Partner' with mode: Specialty Coffee Association
Filling missing values in column 'Harvest.Year' with mode: 2012
Filling missing values in column 'Grading.Date' with mode: July 11th, 2012
Filling missing values in column 'Owner.1' with mode: Juan Luis Alvarado Romero
Filling missing values in column 'Variety' with mode: Caturra
Filling missing values in column 'Processing.Method' with mode: Washed / Wet
Filling missing values in column 'Aroma' with median: 7.58
Filling missing values in column 'Flavor' with median: 7.58
Filling missing values in column 'Aftertaste' with median: 7.42
Filling missing values in column 'Acidity' with median: 7.58
Filling missing values in column 'Body' with median: 7.5
Filling missing values in column 'Balance' with median: 7.5
Filling missing values in column 'Uniformity' with median: 10.0
Filling missing values in column 'Clean.Cup' with median: 10.0
Filling missing values in column 'Sweetness' with median: 10.0
Filling missing values in column 'Cupper.Points' with median: 7.5
Filling missing values in column 'Total.Cup.Points' with median: 82.5
Filling missing values in column 'Moisture' with median: 0.11
Filling missing values in column 'Category.One.Defects' with median: 0.0
Filling missing values in column 'Quakers' with median: 0.0
Filling missing values in column 'Color' with mode: Green
Filling missing values in column 'Category.Two.Defects' with median: 2.0
Filling missing values in column 'Expiration' with mode: December 26th, 2014
```

Comment : The Missing Values are replaced with median and mode. Let's check in the Heatmap and numbers of missing values again.

In [257]:
```
sns.heatmap(df.isnull(),yticklabels=False,cmap = 'viridis') # checking the null values in the heatmap
```

Out [257]: <Axes: >



In [259]:
```
df.isna().sum() # checking if the null values have been replaced
```

Out [259]:
```
Species                 0
Owner                   0
Country.of.Origin       0
Farm.Name               0
Mill                    0
Company                 0
Altitude                0
Region                  0
Producer                0
Number.of.Bags          0
Bag.Weight              0
In.Country.Partner      0
Harvest.Year            0
Grading.Date            0
Owner.1                 0
Variety                 0
Processing.Method       0
Aroma                   0
Flavor                  0
Aftertaste              0
Acidity                 0
Body                    0
Balance                 0
Uniformity              0
Clean.Cup               0
Sweetness               0
Cupper.Points           0
Total.Cup.Points        0
Moisture                0
Category.One.Defects    0
Quakers                 0
Color                   0
Category.Two.Defects    0
Expiration              0
dtype: int64
```

Result

- We can see here that there are no null values now in the data and it was replaced/imputated by Median and Mode of the columns according to their datatype.

➽ Addressing Inconsistencies

└➤ Standardizing and Converting "Bag.Weight" Column into numerical column for missing value replacement

In [163]:
```
display(Markdown(bold_text6))
bf1
```

Out [163]:
```
<IPython.core.display.Markdown object>
0       60 kg
1       60 kg
2           1
3       60 kg
4       60 kg
        ...
1334     2 kg
1335     2 kg
1336     1 kg
1337    5 lbs
1338    5 lbs
Name: Bag.Weight, Length: 1339, dtype: object
```

In [165]:
```
display(Markdown(bold_text7))
af1
```

Out [165]:
```
<IPython.core.display.Markdown object>
0       60.00000
1       60.00000
2        1.00000
3       60.00000
4       60.00000
         ...
1334     2.00000
1335     2.00000
1336     1.00000
1337     2.26796
1338     2.26796
Name: Bag.Weight, Length: 1339, dtype: float64
```

└➤ Standardizing the "Harvest.Year" column for missing value replacement and further analysis.

In [181]:
```
display(Markdown(bold_text6))
bf2
```

Out [181]:
```
<IPython.core.display.Markdown object>
2                        NaN
3                       2014
4                       2014
5                       2013
6                       2012
7                     Mar-10
8                     Mar-10
9                       2014
10                      2014
11                      2014
12                      2014
13    Sept 2009 - April 2010
14                    Mar-10
15                      2014
16                May-August
17                 2009/2010
18                      2015
Name: Harvest.Year, dtype: object
```

In [183]:
```
display(Markdown(bold_text7))
af2
```

Out [183]:
```
<IPython.core.display.Markdown object>
2     None
3     2014
4     2014
5     2013
6     2012
7     2010
8     2010
9     2014
10    2014
11    2014
12    2014
13    2009
14    2010
15    2014
16    None
17    2009
18    2015
Name: Harvest.Year, dtype: object
```

└➤ Standardizing the "Altitude" column for missing value replacement and further analysis.

In [208]:
```
display(Markdown(bold_text6))
bf3
```

Out [208]:
```
<IPython.core.display.Markdown object>
0           1950-2200
1           1950-2200
2       1600 - 1800 m
3           1800-2200
4           1950-2200
            ...
```

```
1334              NaN
1335               40
1336       795 meters
1337              NaN
1338              NaN
Name: Altitude, Length: 1339, dtype: object
```

In [210]:
```
display(Markdown(bold_text7))
af3
```

```
<IPython.core.display.Markdown object>
0       2075.0
1       2075.0
2       1700.0
3       2000.0
4       2075.0
         ...
1334       NaN
1335      40.0
1336       NaN
1337       NaN
1338       NaN
Name: Altitude, Length: 1339, dtype: float64
```

Out [210]:

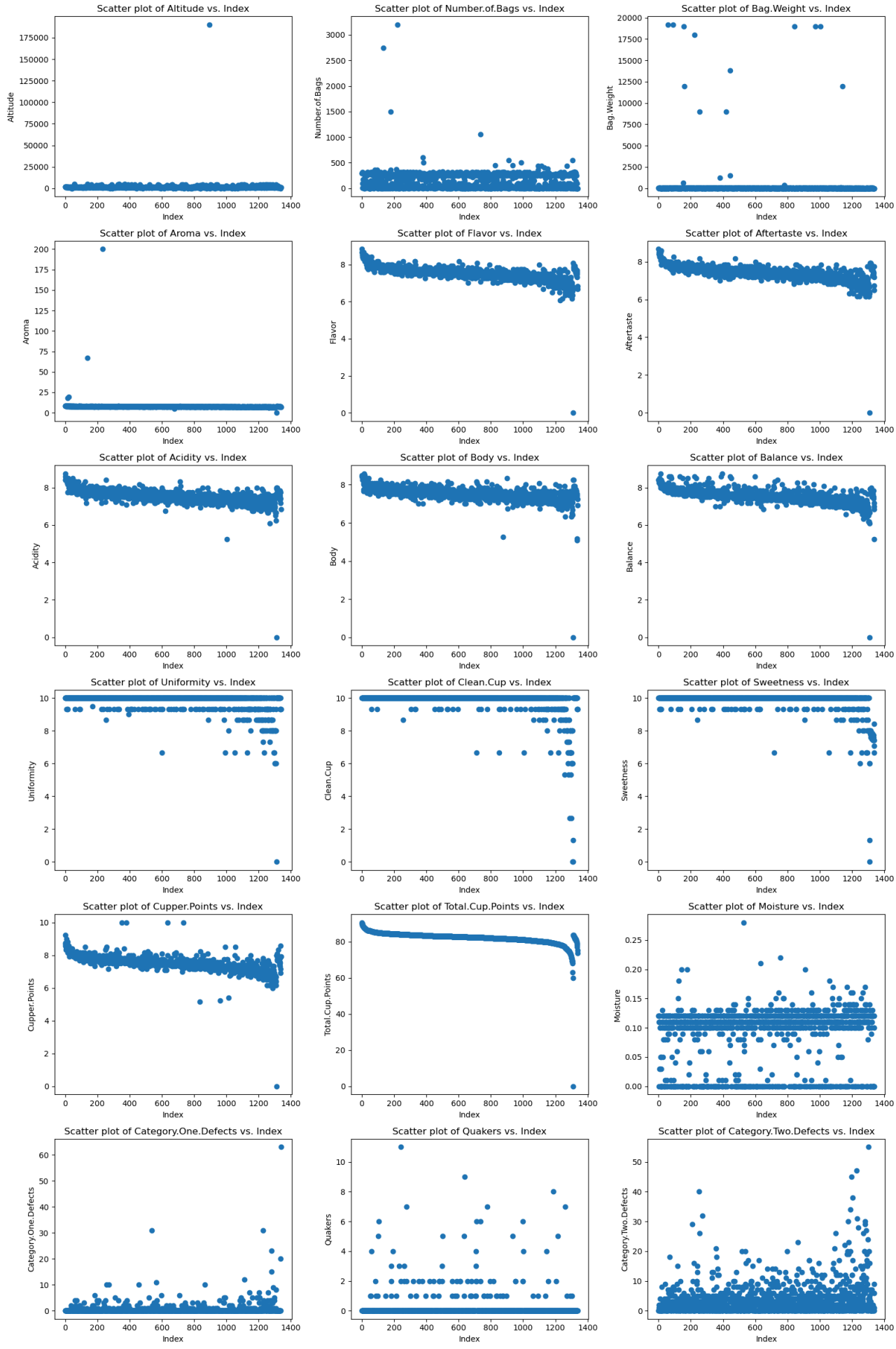
Comment : As we can see here that the Columns "Bag.Weight", "Harvest.Year" and "Altitude" is Standardized. Now we can Imputate/Replace Missing Values.

➽ Outliers

↳➤ Identifying the Outliers

In [559]:
```
display(Markdown(bold_text8))
# Identify numerical columns
numerical_columns = df.select_dtypes(include='number').columns
# Determine the number of rows and columns for the grid
n_cols = 3
n_rows = int(np.ceil(len(numerical_columns) / n_cols))
# Create subplots
fig, axes = plt.subplots(n_rows, n_cols, figsize=(16, n_rows * 4))
axes = axes.flatten()
# Plot scatter plots in the grid
for idx, col in enumerate(numerical_columns):
    ax = axes[idx]
    ax.scatter(df.index, df[col])
    ax.set_title(f'Scatter plot of {col} vs. Index')
    ax.set_xlabel('Index')
    ax.set_ylabel(col)
# Remove any empty subplots
for i in range(len(numerical_columns), len(axes)):
    fig.delaxes(axes[i])

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

```
<IPython.core.display.Markdown object>
```



Comment :

- From the above ScatterPlot we can able to see the outliers present in the data.
- The Category.Two.Defects havethe most number of outliers in the Dataset

Z-Score Test

Using zscore to find the Outliers in the important columns for Anaysis

In [572]:
```
display(Markdown(bold_text9))
print('✔️ Number.of.Bags :',A)
print('✔️ Bag.Weight :',B)
print('✔️ Altitude :',C)
print('✔️ Aroma :',D)
print('✔️ Flavor :',E)
print('✔️ Aftertaste :',F)
print('✔️ Acidity :',G)
print('✔️ Clean.Cup :',H)
print('✔️ Category.One.Defects :',I)
print('✔️ Category.Two.Defects :',J)
```

```
<IPython.core.display.Markdown object>

✔️ Number.of.Bags : (array([133, 179, 220, 734], dtype=int64),)
✔️ Bag.Weight : (array([  59,   88,  158,  159,  224,  255,  420,  443,  843,  974, 1005,
       1139], dtype=int64),)
✔️ Altitude : (array([896], dtype=int64),)
✔️ Aroma : (array([139, 234], dtype=int64),)
✔️ Flavor : (array([   0, 1230, 1245, 1304, 1310], dtype=int64),)
✔️ Aftertaste : (array([   0, 1230, 1245, 1283, 1284, 1289, 1303, 1304, 1305, 1310],
      dtype=int64),)
✔️ Acidity : (array([   0, 1002, 1269, 1308, 1310], dtype=int64),)
✔️ Clean.Cup : (array([ 712,  851, 1006, 1171, 1219, 1259, 1270, 1277, 1278, 1279, 1280,
       1282, 1285, 1288, 1289, 1290, 1291, 1295, 1296, 1297, 1298, 1299,
       1300, 1301, 1302, 1306, 1307, 1308, 1309, 1310], dtype=int64),)
✔️ Category.One.Defects : (array([ 256,  271,  457,  536,  564,  866, 1113, 1229, 1279, 1280, 1291,
       1337, 1338], dtype=int64),)
✔️ Category.Two.Defects : (array([ 209,  250,  256,  271,  357,  518,  537,  797,  864, 1091, 1097,
       1154, 1172, 1176, 1179, 1188, 1190, 1195, 1203, 1229, 1232, 1239,
       1269, 1276, 1277, 1279, 1282, 1289, 1299, 1302, 1306], dtype=int64),)
```

➥ Removal of Outliers

This is the final step of Data Cleaning which is removal of Outliers present in the Data to ensure and maintain the accuracy of the Data and Insigths drived from it.

In [624]:
```
display(Markdown(bold_text10))
print("Shape of the Data Outlier Before Outlier Removal :",bfotl)
print("Shape of the Data Outlier After Outlier Removal :",afotl)
```

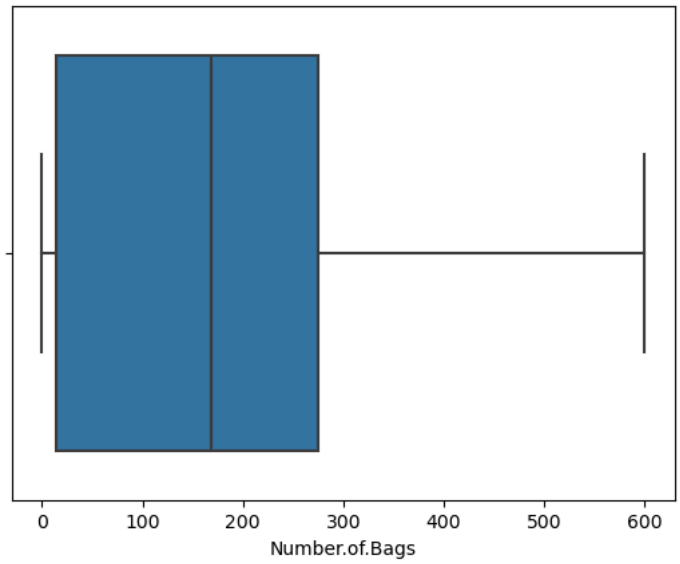
```
<IPython.core.display.Markdown object>

Shape of the Data Outlier Before Outlier Removal : (1339, 34)
Shape of the Data Outlier After Outlier Removal : (1312, 35)
```

Results

- 27 Index Rows has been droped from the df by removal of Outliers from the data.
- For Example you can see the BoxPlot below that there no outliers after removing.

In [629]:
```
sns.boxplot(x=df2["Number.of.Bags"])
```

Out [629]:
```
<Axes: xlabel='Number.of.Bags'>
```



7) Obtaining Derived Metrics :

➢ *Description of derived metrics created to enhance the dataset.*

In [668]:
```
df2.head()
```

Out [668]:

	Species	Owner	Country.of.Origin	Farm.Name	Mill	Company	Altitude	Region	Producer	Number.of.Bags	...	Expiration
0	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural	2075.0	guji-hambela	METAD PLC	300.0	...	April 3rd, 2016

	Species	Owner	Country.of.Origin	Farm.Name	Mill	Company	Altitude	Region	Producer	Number.of.Bags	...	Expiration
						developmet plc						
1	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	April 3rd, 2016
2	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	beneficio ixchel	unex guatemala, s.a.	1700.0	huila	La Plata	5.0	...	May 31st, 2011
3	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	wolensu	yidnekachew debessa coffee plantation	2000.0	oromia	Yidnekachew Dabessa Coffee Plantation	320.0	...	March 25th, 2016
4	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	April 3rd, 2016

5 rows × 43 columns

In [680]:
```
# Summary statistics of the new derived metrics
display(Markdown(bold_text11))
print(df2[['Total_Quality_Score', 'Aroma_to_Flavor_Ratio'] + [f'Normalized_{col}' for col in quality_columns]]
```

```
<IPython.core.display.Markdown object>

       Total_Quality_Score  Aroma_to_Flavor_Ratio  Normalized_Aroma  \
count          1312.000000            1312.000000      1.312000e+03
mean             45.113742               1.008812     -1.949660e-16
std               1.780924               0.057805      1.000000e+00
min              38.680000               0.655484     -4.704829e+00
25%              44.090000               0.989333     -3.182142e-01
50%              45.170000               1.000000     -1.827470e-02
75%              46.080000               1.022667      3.004110e-01
max              59.790000               2.350061      2.176483e+01

       Normalized_Flavor  Normalized_Aftertaste  Normalized_Acidity  \
count       1.312000e+03           1.312000e+03        1.312000e+03
mean       -9.531671e-16           1.213122e-15        1.819683e-15
std         1.000000e+00           1.000000e+00        1.000000e+00
min        -4.248156e+00          -3.543280e+00       -7.184642e+00
25%        -5.783715e-01          -4.514389e-01       -6.635418e-01
50%         1.555854e-01           3.523969e-02        1.202443e-01
75%         6.546761e-01           4.932901e-01        6.532188e-01
max         3.825370e+00           3.613759e+00        3.788363e+00

       Normalized_Body  Normalized_Balance
count     1.312000e+03        1.312000e+03
mean      1.126470e-15       -1.603054e-15
std       1.000000e+00        1.000000e+00
min      -7.926367e+00       -6.442797e+00
25%      -6.257690e-01       -5.517069e-01
50%      -7.416833e-02       -7.022352e-02
75%       4.774324e-01        6.378402e-01
max       3.430119e+00        3.470095e+00
```

==========

8) Filtering Data for Analysis :

- *Explanation of any additional data filtering or preprocessing steps performed to prepare the data for analysis.*

In [686]:
```
quality_measures = df2[['Aroma','Flavor','Aftertaste','Acidity','Body','Balance','Uniformity','Cupper.Points',
                        'Total.Cup.Points','Quakers','Category.One.Defects','Category.Two.Defects']]
bean_metadata = df2[['Species','Processing.Method','Color']]
farm_metadata = df2[['Owner','Country.of.Origin','Farm.Name','Mill','Company','Altitude','Region']]
```

Further Classifying the Columns based on the Type of Data it has

NOTE:

This filtering will help during the Chi-Square. lets check which columns are Categorical Data in Non-Numeric Columns. Iam going to further classify the Non_Numerical_Data into Categorical_Data, Unique_Data, and Date_Data.

- The Columns which has less than 100 unique values are conisdered as Categorical_Data.
- The Columns which has more than 100 unique values are conisdered as Unique_Data.
- The Columns which has date values in it are Date_Data.

In [702]:
```
# Check for the count of unique values in each categorical column
unique_counts = {}
for col in Non_Numerical_Data:
    unique_count = df2[col].nunique()
    unique_counts[col] = unique_count
    print("Count of unique values in " + col + ": " + str(unique_count))
```

```
Count of unique values in Species: 2
Count of unique values in Owner: 309
```

```
Count of unique values in Country.of.Origin: 36
Count of unique values in Farm.Name: 563
Count of unique values in Mill: 451
Count of unique values in Company: 276
Count of unique values in Altitude: 173
Count of unique values in Region: 349
Count of unique values in Producer: 680
Count of unique values in Bag.Weight: 45
Count of unique values in In.Country.Partner: 26
Count of unique values in Harvest.Year: 10
Count of unique values in Grading.Date: 556
Count of unique values in Owner.1: 313
Count of unique values in Variety: 29
Count of unique values in Processing.Method: 5
Count of unique values in Color: 3
Count of unique values in Expiration: 555
```

In [706]:
```
# Classify columns
categorical_data = []
unique_data = []
date_data = ['Grading.Date','Expiration','Harvest.Year']  # Initially assuming 'Grading.Date' as a date column

for col, count in unique_counts.items():
    if count < 100:
        categorical_data.append(col)
    elif col in date_data:
        continue
    else:
        unique_data.append(col)

# Assuming columns that have 'Date' or 'Year' in their name are date columns
date_data += [col for col in df2[Non_Numerical_Data].columns if 'Date' in col or 'Year' in col]

# Remove date columns from unique_data if they were added by mistake
unique_data = [col for col in unique_data if col not in date_data]

# Print the classified columns
print("Categorical_Data: ", categorical_data)
print(" ")
print("Unique_Data: ", unique_data)
print(" ")
print("Date_Data: ", date_data)
```

```
Categorical_Data:  ['Species', 'Country.of.Origin', 'Bag.Weight', 'In.Country.Partner', 'Harvest.Year', 'Variety', 'Processing.Method', 'Color']

Unique_Data:  ['Owner', 'Farm.Name', 'Mill', 'Company', 'Altitude', 'Region', 'Producer', 'Owner.1']

Date_Data:  ['Grading.Date', 'Expiration', 'Harvest.Year', 'Harvest.Year', 'Grading.Date']
```

In [714]:
```
Numericals_Data = df2[['Number.of.Bags', 'Bag.Weight', 'Altitude','Aroma', 'Flavor', 'Aftertaste','Acidity', 'I
                       'Clean.Cup', 'Sweetness','Cupper.Points', 'Total.Cup.Points', 'Moisture', 'Category.One
Categorical_Data = df2[['Country.of.Origin', 'In.Country.Partner', 'Variety', 'Species', 'Processing.Method',
Date_Data = df2[['Grading.Date', 'Expiration', 'Harvest.Year']]
Unique_Data = df2[['Owner', 'Farm.Name', 'Mill', 'Company', 'Region', 'Producer']]
```

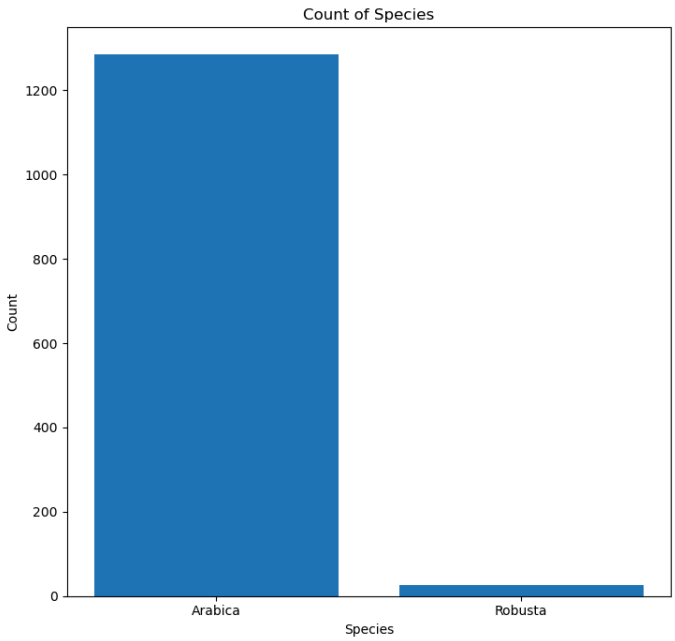
==========

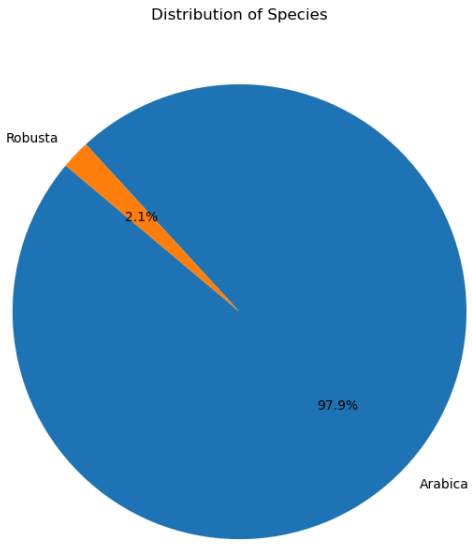
9) EDA - Univariate Analysis :

- *Insights gained from univariate analysis, including visualizations of individual variables.*

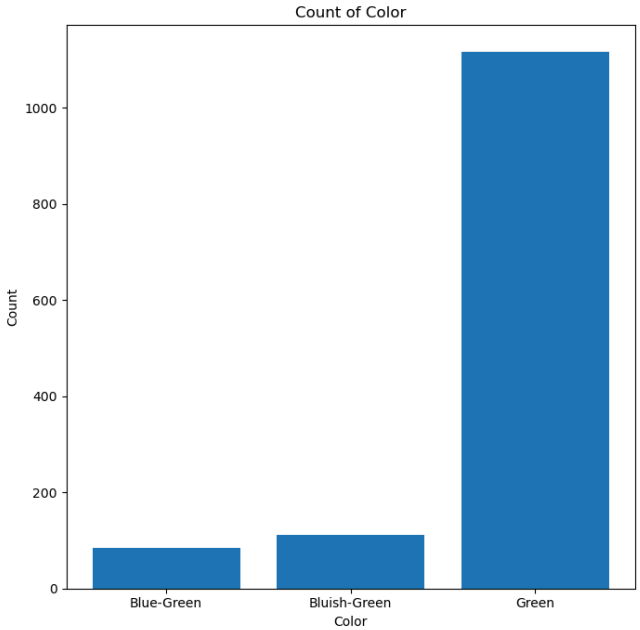
Visualisation of Distribution of Species, Color and Processing.Method of Coffee by Bar Chart and Pie Chart.

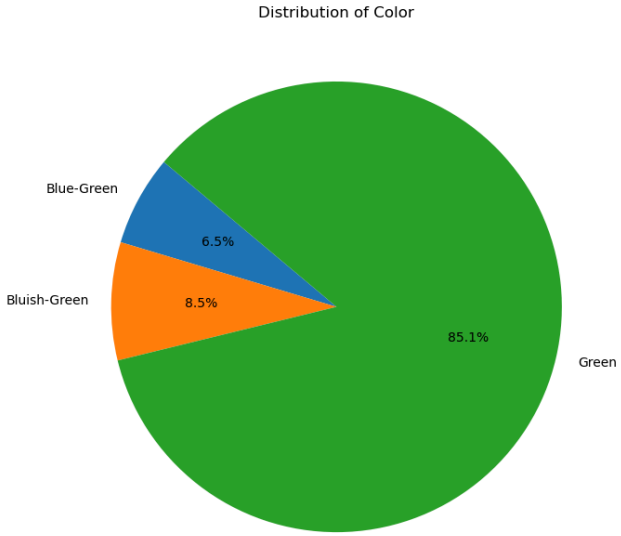
In [827]:
```
# Creating a figure with two subplots: one for the bar chart and one for the pie chart
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Bar chart
axs[0].bar(fre_tab['Species'], fre_tab['Count'])
axs[0].set_title('Count of Species')
axs[0].set_xlabel('Species')
axs[0].set_ylabel('Count')
# Pie chart
axs[1].pie(fre_tab['Count%'], labels=fre_tab['Species'], autopct='%1.1f%%', startangle=140)
axs[1].set_title('Distribution of Species')
# Display the charts
plt.tight_layout()
plt.show()
```



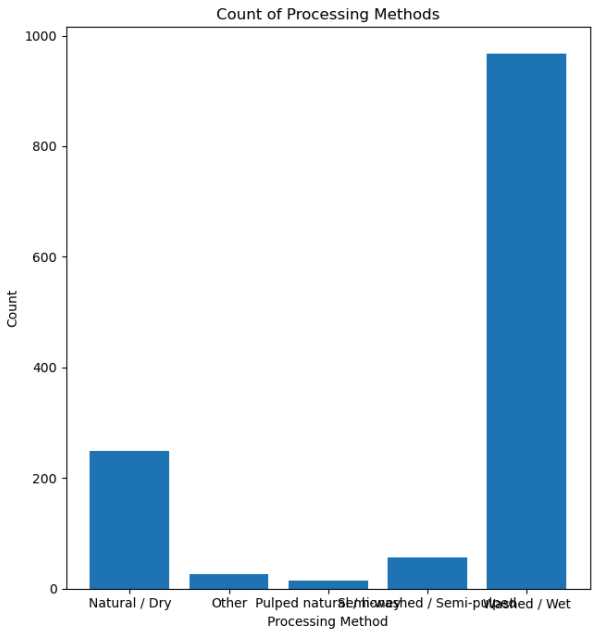


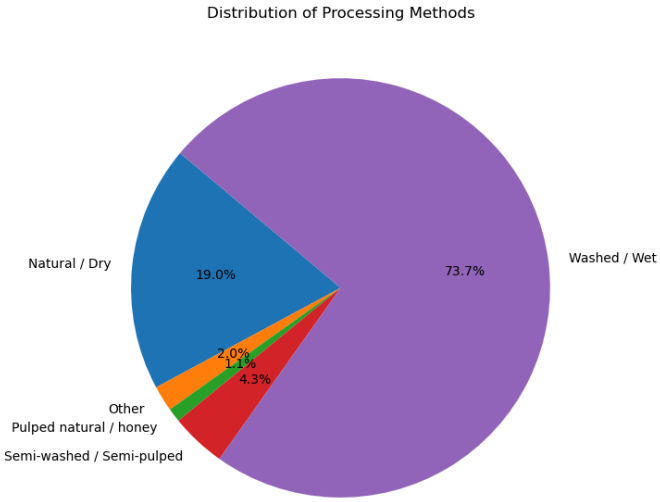
In [829]:
```
# Create a figure with two subplots: one for the bar chart and one for the pie chart
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Bar chart
axs[0].bar(fre_tab1['Color'], fre_tab1['Count'])
axs[0].set_title('Count of Color')
axs[0].set_xlabel('Color')
axs[0].set_ylabel('Count')
# Pie chart
axs[1].pie(fre_tab1['Count%'], labels=fre_tab1['Color'], autopct='%1.1f%%', startangle=140)
axs[1].set_title('Distribution of Color')
# Display the charts
plt.tight_layout()
plt.show()
```





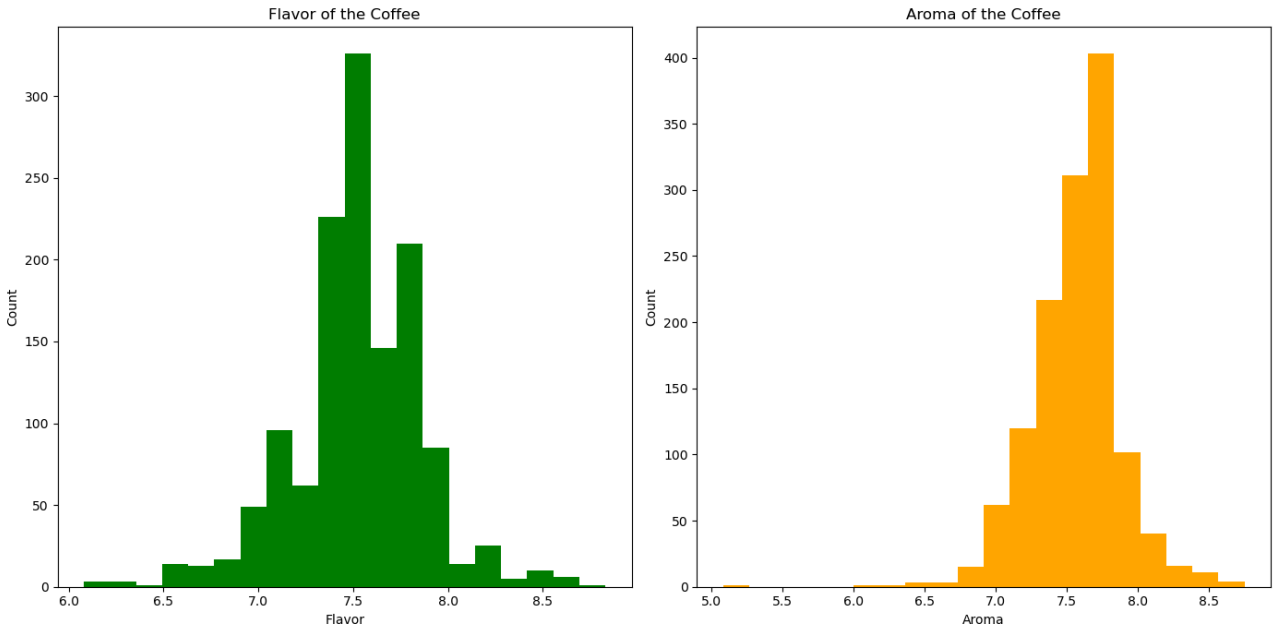
In [831]:
```
# Create a figure with two subplots: one for the bar chart and one for the pie chart
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Bar chart
axs[0].bar(fre_tab2['Processing.Method'], fre_tab2['Count'])
axs[0].set_title('Count of Processing Methods')
axs[0].set_xlabel('Processing Method')
axs[0].set_ylabel('Count')
# Pie chart
axs[1].pie(fre_tab2['Count%'], labels=fre_tab2['Processing.Method'], autopct='%1.1f%%', startangle=140)
axs[1].set_title('Distribution of Processing Methods')
# Display the charts
plt.tight_layout()
plt.show()
```





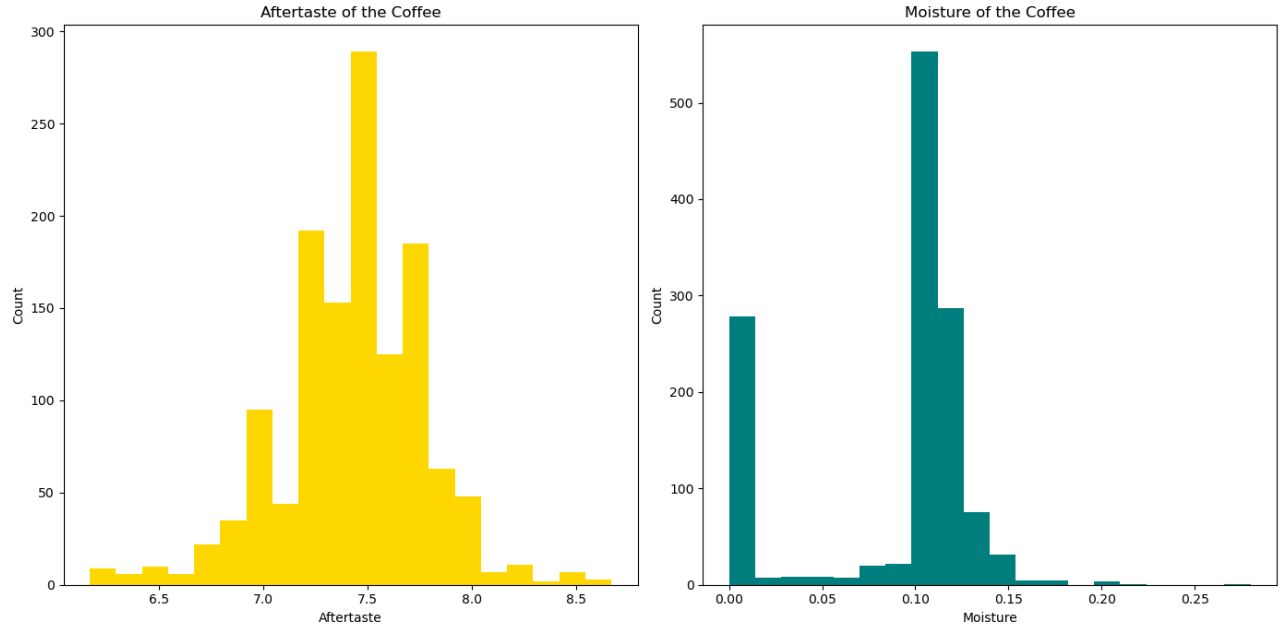
Visualisation of Count Distributions of Quality Measures of Coffee :

In [835]:
```
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Plot the histogram for the 'Flavor' values
x = df2["Flavor"]
axs[0].hist(x, bins=20, color="green")
axs[0].set_title("Flavor of the Coffee")
axs[0].set_xlabel("Flavor")
axs[0].set_ylabel("Count")
# Plot the histogram for the 'Aroma' values
y = cleaned_df["Aroma"]
axs[1].hist(y, bins=20, color="orange")
axs[1].set_title("Aroma of the Coffee")
axs[1].set_xlabel("Aroma")
axs[1].set_ylabel("Count")
# Display the charts
plt.tight_layout()
plt.show()
```



In [837]:
```
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Plot the histogram for the 'Flavor' values
x = df2["Aftertaste"]
axs[0].hist(x, bins=20, color="gold")
axs[0].set_title("Aftertaste of the Coffee")
axs[0].set_xlabel("Aftertaste")
axs[0].set_ylabel("Count")
# Plot the histogram for the 'Aroma' values
y = df2["Moisture"]
axs[1].hist(y, bins=20, color="Teal")
axs[1].set_title("Moisture of the Coffee")
```

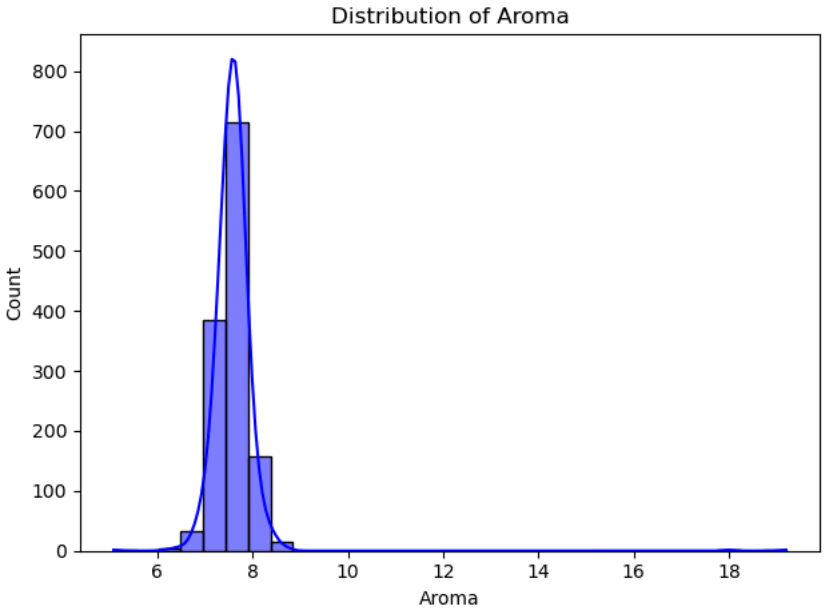
```
axs[1].set_xlabel("Moisture")
axs[1].set_ylabel("Count")
# Display the charts
plt.tight_layout()
plt.show()
```



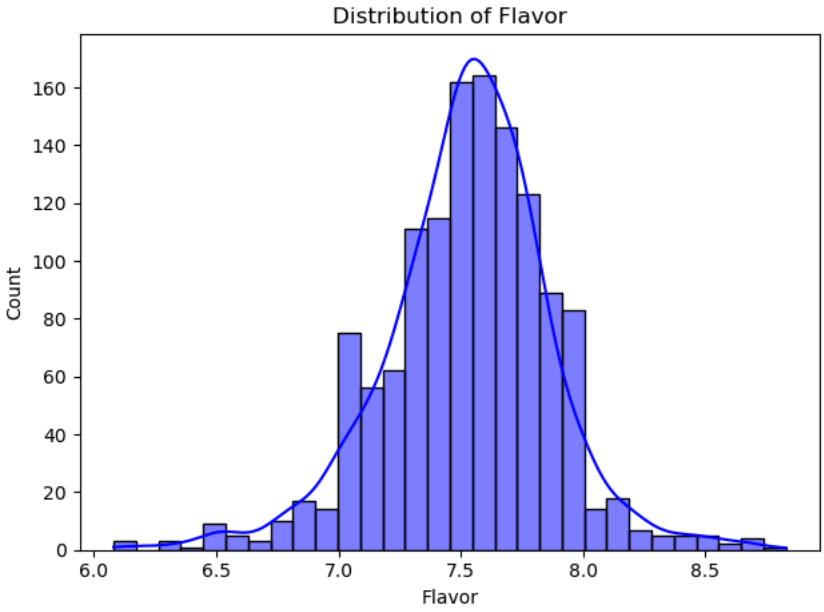
Distributions

In [840]:
```
# Plot histograms for quality measures
for measure in quality_measures:
    plt.figure()
    sns.histplot(df2[measure], kde=True, color='blue', bins=30)
    plt.title(f'Distribution of {measure}')
    plt.tight_layout()
    plt.show()
```

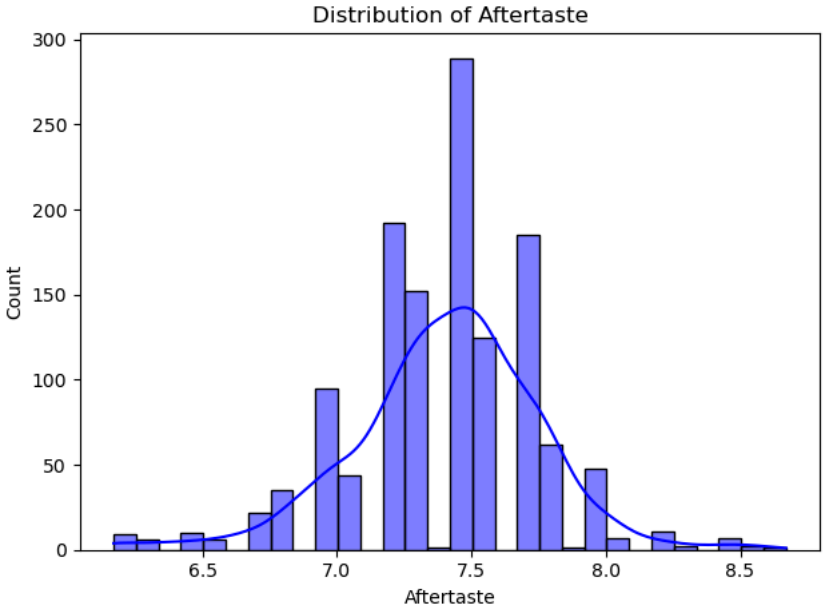
```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



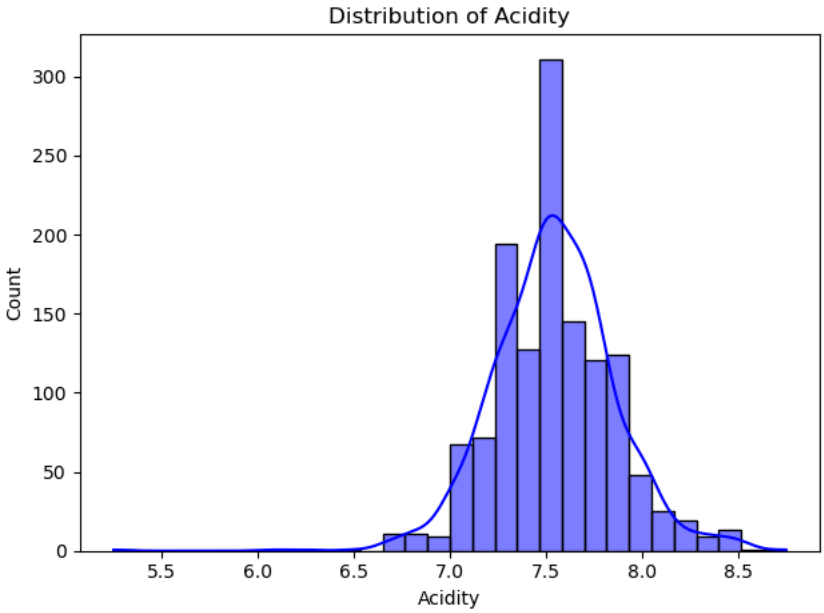

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



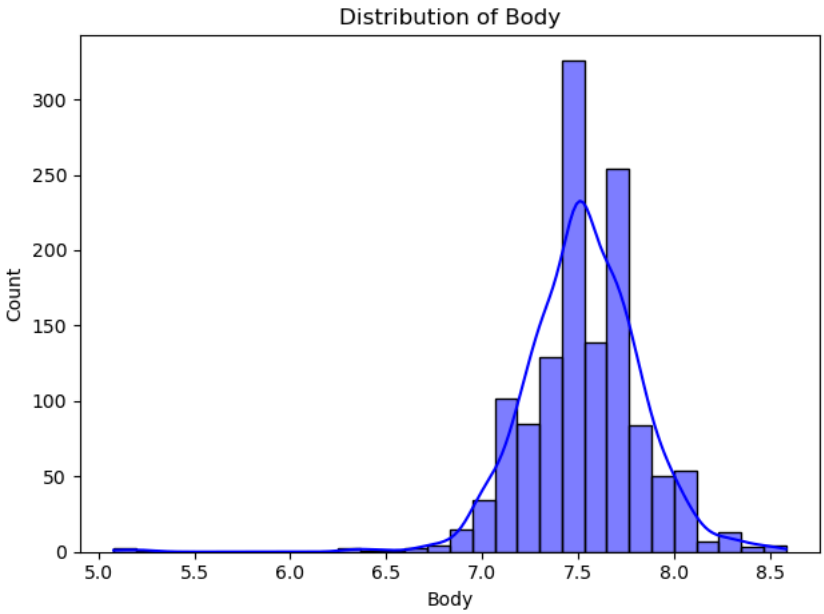

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



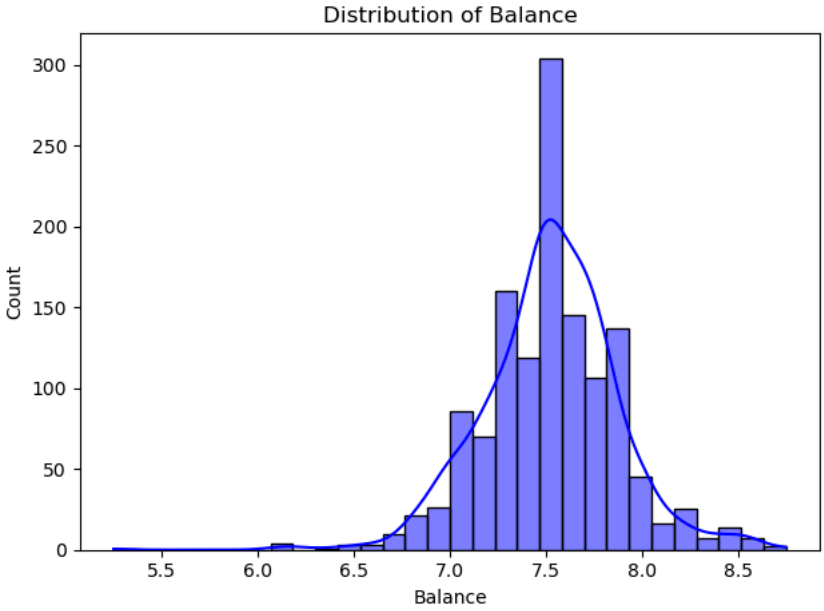

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



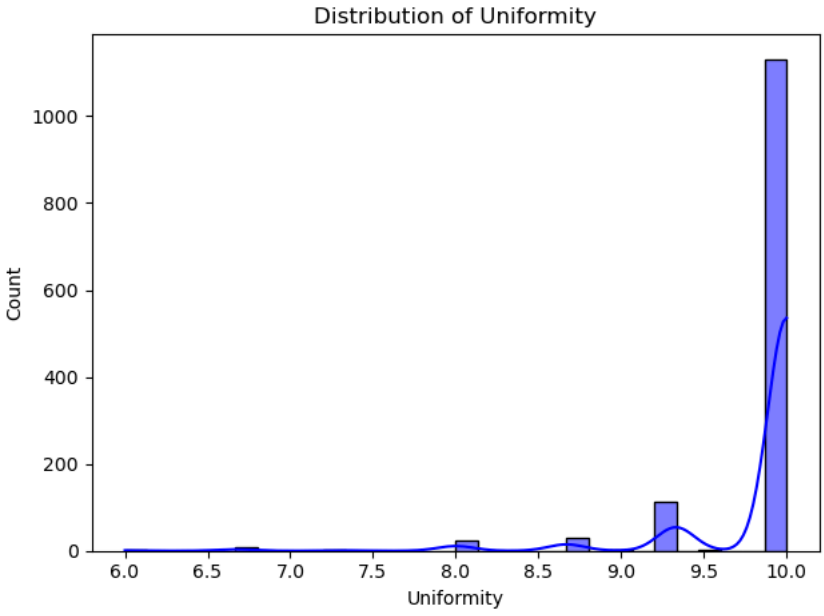

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



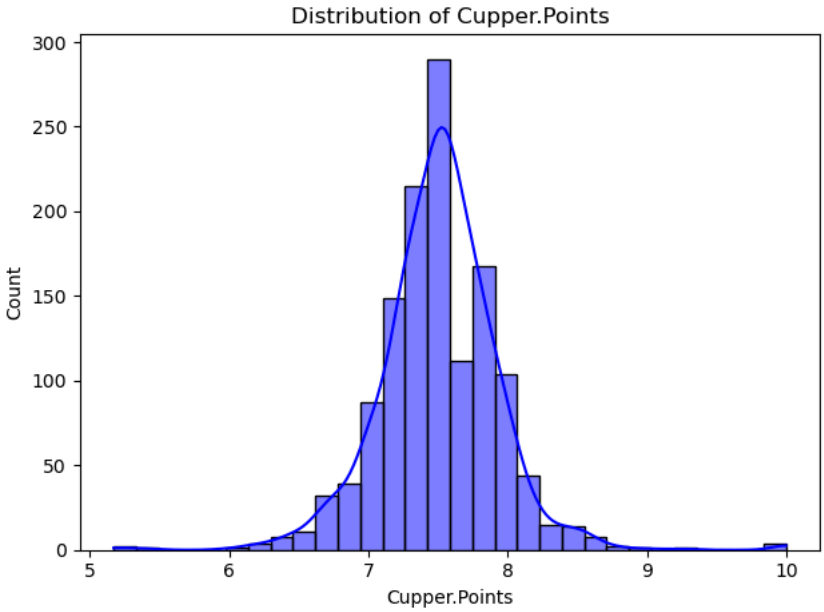

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



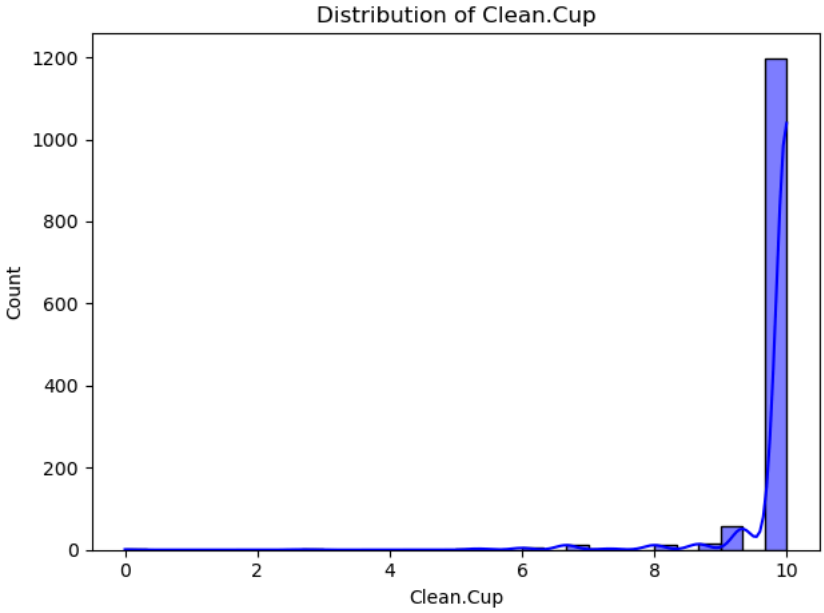

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



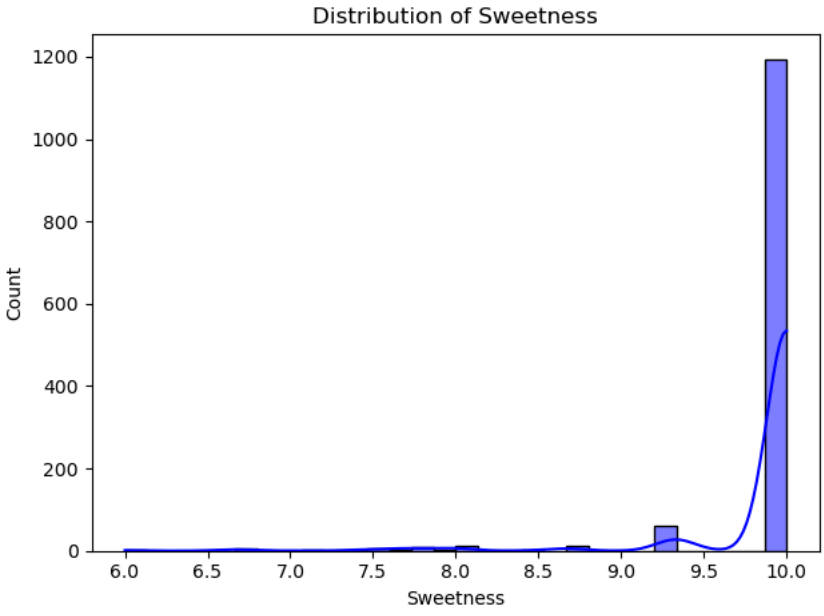

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



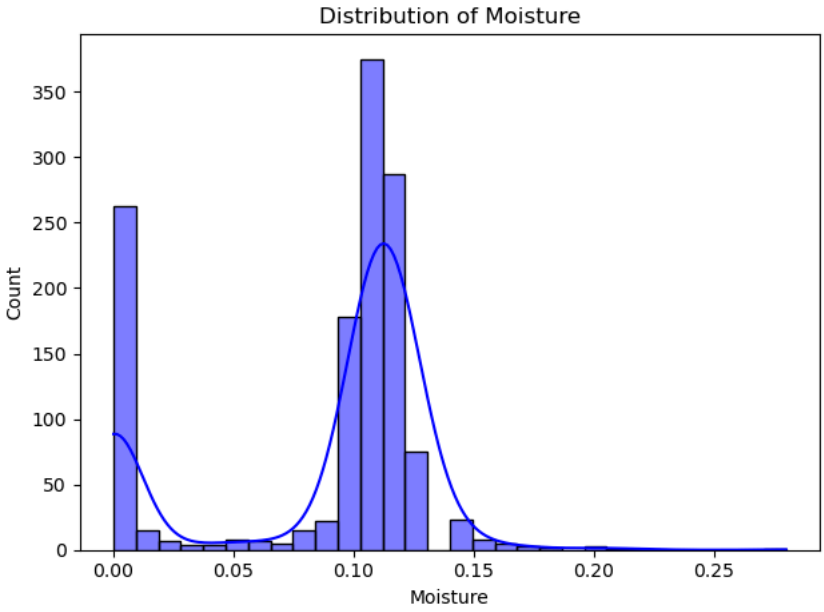

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



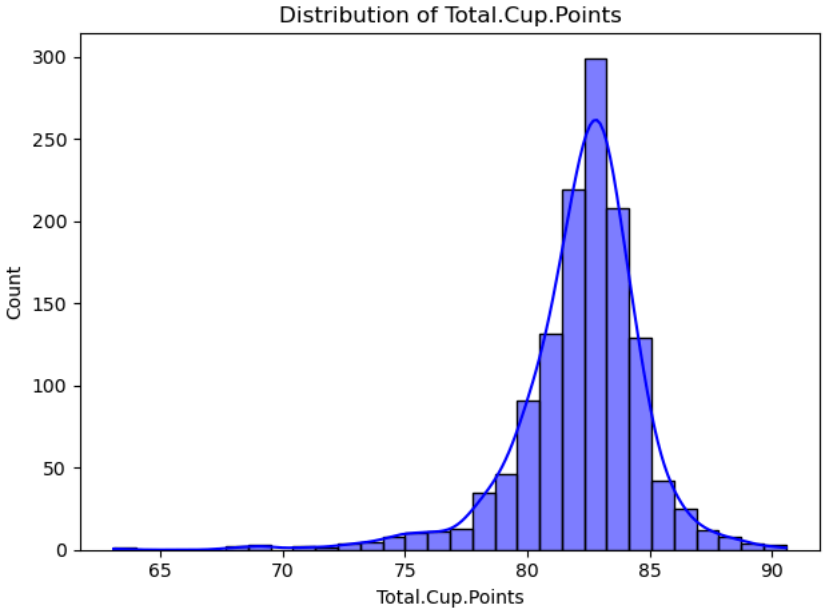

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



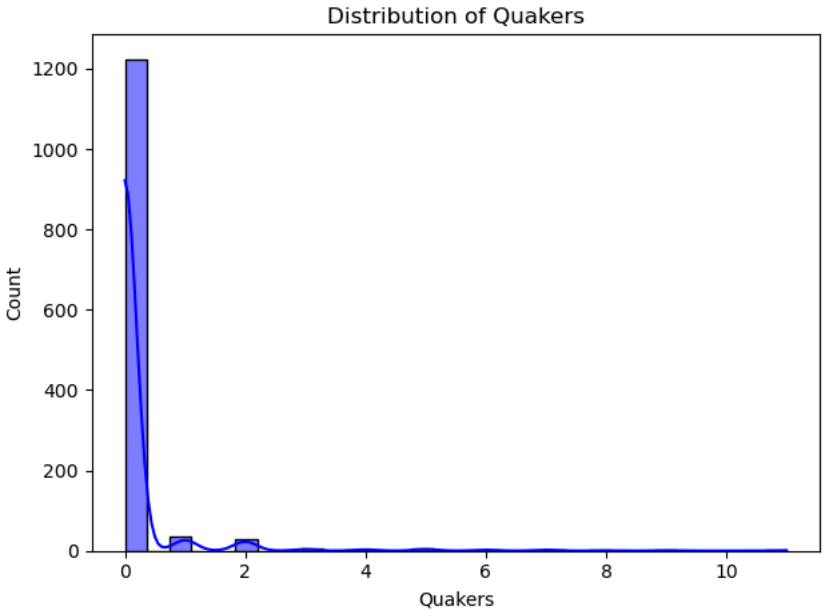

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



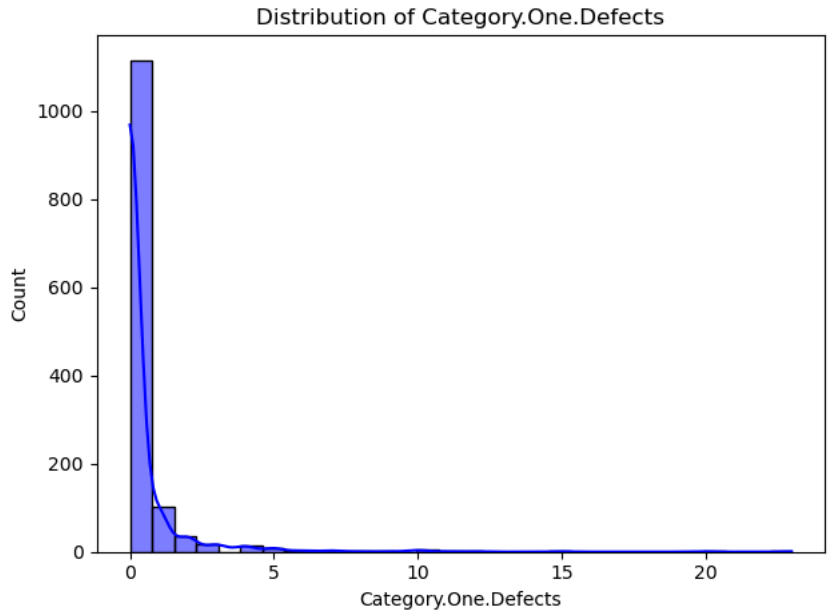

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



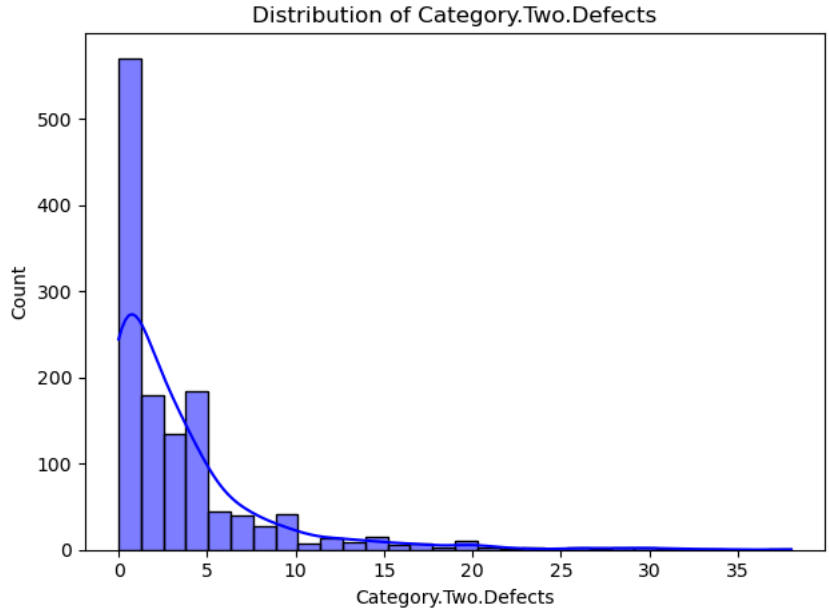

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Insights gained from Univariate Analysis :

- **Acidity:** There's a possible normal distribution (bell-shaped curve) indicating most responses fall around the average acidity level.
- **Body:** There's a possible right-skewed distribution suggesting more ratings towards a lighter body.
- **Balance and Uniformity:** The results are inconclusive due to missing labels on the axes.
- **Other Features:** There were mentions of factors like Cupper Points, Clean Cup score, sweetness, moisture, and Quakers (which likely meant something else taste-related). However, the exact details and interpretations were unclear due to missing labels or potential data entry errors.

Overall fromthe chart it suggests the coffee might be on the lighter side in terms of acidity and body.
Here are some additional questions that could be answered with more information:

- What roast profile was used for the coffee?
- What brewing method was used?
- What is the origin of the coffee beans?
- What do the labels for the axes in the graphs represent? Knowing these details would allow for a more comprehensive analysis of the coffee.

==========

10) Segmented Univariate Analysis :

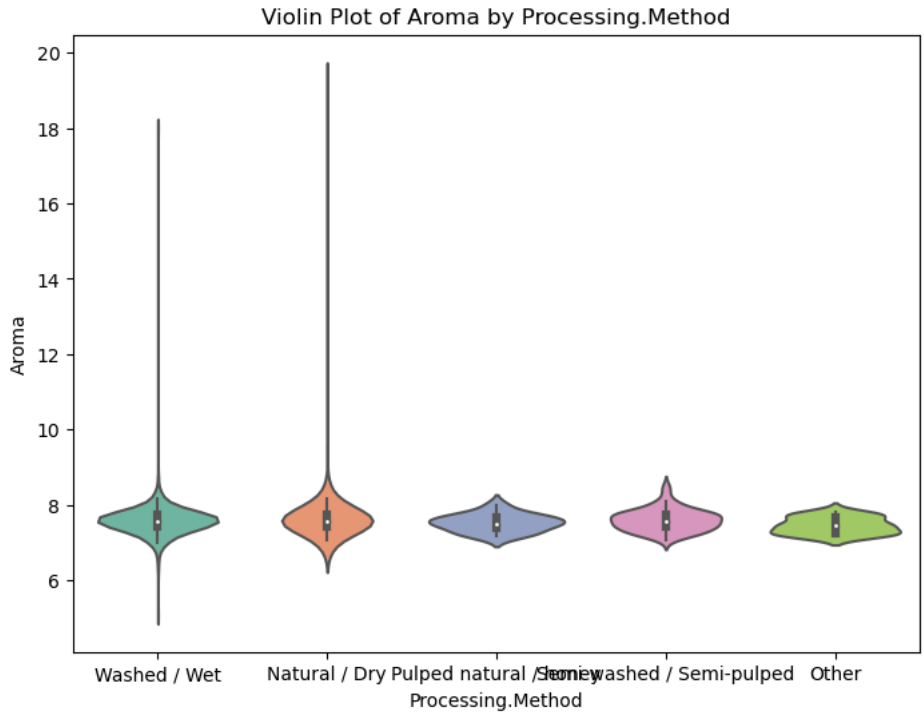
- *Analysis of data segments or categories to gain deeper insights.*

In [843]:

```
# Define a function to plot segmented univariate analysis for numerical variables
def plot_segmented_univariate_analysis(df2, numerical_column, segment_column):
    plt.figure(figsize=(18, 6))
    # Violin Plot
    plt.subplot(1, 2, 2)
    sns.violinplot(x=segment_column, y=numerical_column, data=df2, palette='Set2')
    plt.title(f'Violin Plot of {numerical_column} by {segment_column}')
    plt.xlabel(segment_column)
    plt.ylabel(numerical_column)
```

```
    plt.show()

# Perform segmented univariate analysis for 'Aroma' across different 'Processing Method'
print("\nSegmented analysis for 'Aroma' by 'Processing Method':")
plot_segmented_univariate_analysis(df2, 'Aroma', 'Processing.Method')
```

```
Segmented analysis for 'Aroma' by 'Processing Method':
```



In [896]:

```
import plotly.express as px
# Check if the required columns are present
required_columns = ['Aroma', 'Flavor', 'Acidity']
if all(column in df2.columns for column in required_columns):
    # Filter the dataframe to include only the required columns and drop any rows with missing values
    df_filtered = df2[required_columns + ['Processing.Method']].dropna()

    # Create 3D scatter plot
    fig = px.scatter_3d(df_filtered,
                        x='Aroma',
                        y='Flavor',
                        z='Acidity',
                        color='Processing.Method',  # Use the correct column name here
                        title='3D Scatter Plot of Aroma, Flavor, and Acidity by Processing Method',
                        labels={'Aroma': 'Aroma Score', 'Flavor': 'Flavor Score', 'Acidity': 'Acidity Score'})

    # Show the plot
    fig.show()
else:
    print("The required columns are not present in the dataframe.")
```

Insights gained from Segmented Univariate Analysis :

- The provided screenshot includes two main parts: a violin plot of the "Aroma" variable segmented by "Processing Method" and a correlation matrix. Here are the insights derived from each:

Insights from the Violin Plot

The violin plot illustrates the distribution of the "Aroma" scores across different "Processing Methods". The key observations are:

✨**Washed / Wet Method:**

- This method shows the highest median Aroma score among the methods displayed.
- The distribution is wide, indicating a high variability in Aroma scores.

✨ **Natural / Dry Method:**

- The median Aroma score for this method is lower than the Washed/Wet method.
- It also shows a wide distribution, similar to the Washed/Wet method, indicating considerable variability.

✨ **Pulped natural / Semi-washed Method:**

- This method shows a moderate median Aroma score.
- The distribution is more compact compared to the Washed/Wet and Natural/Dry methods, suggesting less variability in Aroma scores.

✨ **Semi-pulped Method:**

- This method shows a lower median Aroma score.
- The distribution is relatively narrow, indicating consistent Aroma scores within this method.

✨ **Other Methods:**

- These methods show the lowest median Aroma scores.
- The distribution is quite narrow, indicating less variability in Aroma scores.

Insights from the Violin Plot

The 3D scatterplot shows the relationship between "Aroma," "Flavor," and "Acidity" scores, with data points color-coded by "Processing Method." The plot allows for the visualization of how these three quality metrics interact across different processing methods. Key observations include:

✨ **Cluster Formation:**

- Different processing methods form distinct clusters, indicating variations in "Aroma," "Flavor," and "Acidity" scores.
  ✨ **Data Distribution :**
- Washed/Wet and Natural/Dry methods seem to dominate certain regions in the plot, suggesting they have different profiles compared to other methods.

==========

11) Bivariate Analysis :

- *Analysis of relationships between pairs of variables.*

➽ Correlations :

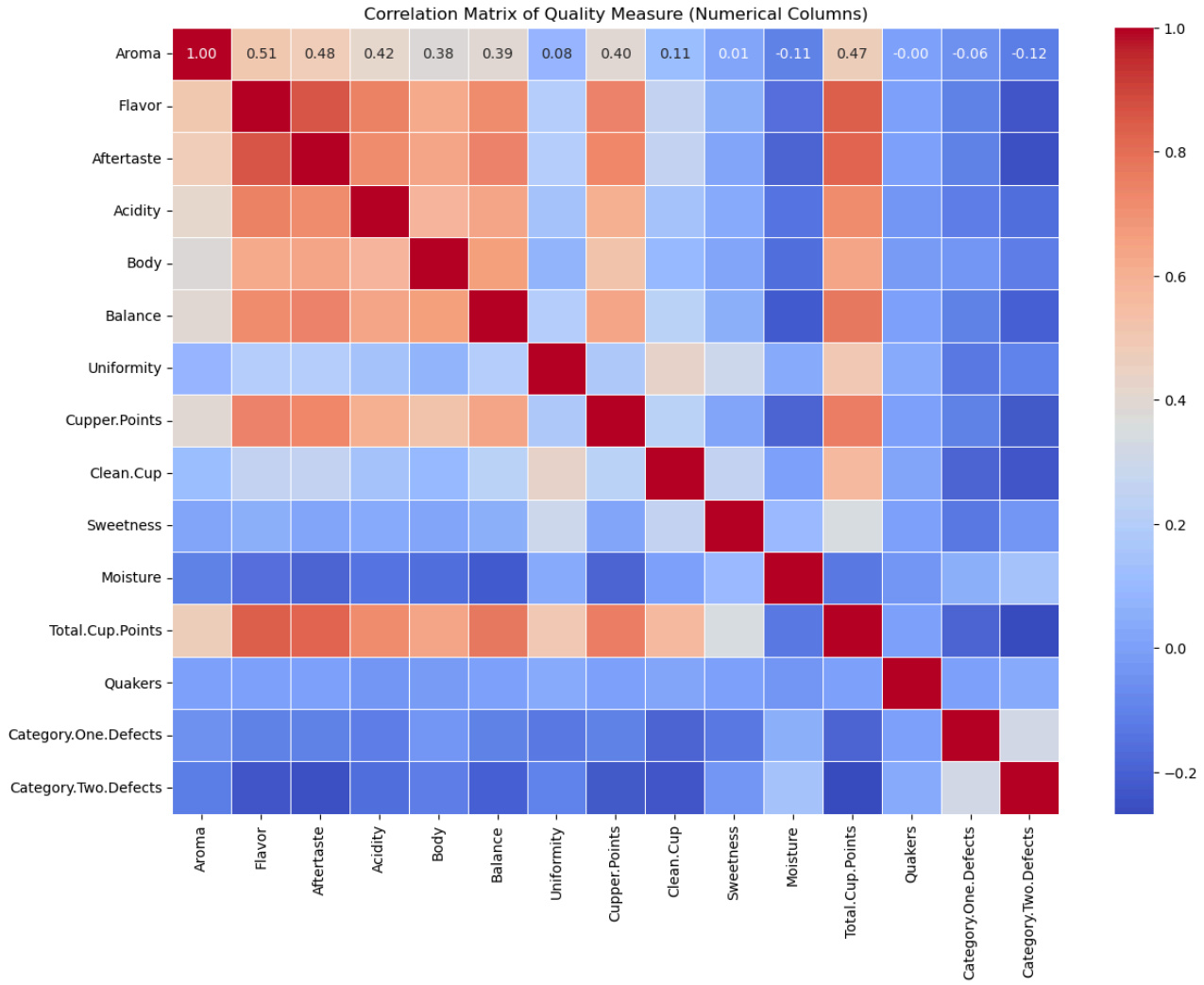
In [1082]:
```
corr
```

Out [1082]:

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Cupper.Points	Clean.Cup	Sweetness	Moist
Aroma	1.000000	0.508439	0.480144	0.416105	0.382436	0.393577	0.084490	0.397754	0.110760	0.011011	-0.105
Flavor	0.508439	1.000000	0.859311	0.745544	0.627433	0.714741	0.201286	0.739345	0.254111	0.050702	-0.156
Aftertaste	0.480144	0.859311	1.000000	0.716186	0.632507	0.742176	0.195325	0.729566	0.251086	0.028843	-0.196
Acidity	0.416105	0.745544	0.716186	1.000000	0.575009	0.630714	0.147355	0.612006	0.135906	0.030904	-0.140
Body	0.382436	0.627433	0.632507	0.575009	1.000000	0.663154	0.071847	0.523495	0.096100	0.019724	-0.164
Balance	0.393577	0.714741	0.742176	0.630714	0.663154	1.000000	0.194029	0.643432	0.222935	0.051131	-0.220
Uniformity	0.084490	0.201286	0.195325	0.147355	0.071847	0.194029	1.000000	0.180527	0.427653	0.298302	0.032
Cupper.Points	0.397754	0.739345	0.729566	0.612006	0.523495	0.643432	0.180527	1.000000	0.226903	0.015596	-0.195
Clean.Cup	0.110760	0.254111	0.251086	0.135906	0.096100	0.222935	0.427653	0.226903	1.000000	0.254107	-0.000
Sweetness	0.011011	0.050702	0.028843	0.030904	0.019724	0.051131	0.298302	0.015596	0.254107	1.000000	0.105
Moisture	-0.105539	-0.156767	-0.196938	-0.140848	-0.164497	-0.220364	0.032522	-0.195375	-0.000214	0.105693	1.0000
Total.Cup.Points	0.473607	0.836144	0.823709	0.715897	0.641770	0.769539	0.499094	0.753961	0.562545	0.338647	-0.138
Quakers	-0.003074	-0.001308	-0.002987	-0.025130	-0.017017	0.004670	0.031915	0.006206	0.027220	0.001681	-0.034
Category.One.Defects	-0.058233	-0.105919	-0.106212	-0.110116	-0.032910	-0.106494	-0.135308	-0.108687	-0.192902	-0.137507	0.047
Category.Two.Defects	-0.121994	-0.231346	-0.243094	-0.167407	-0.111612	-0.204977	-0.100090	-0.224596	-0.228007	-0.040871	0.134

Heatmap for Correlation of Quality Measures

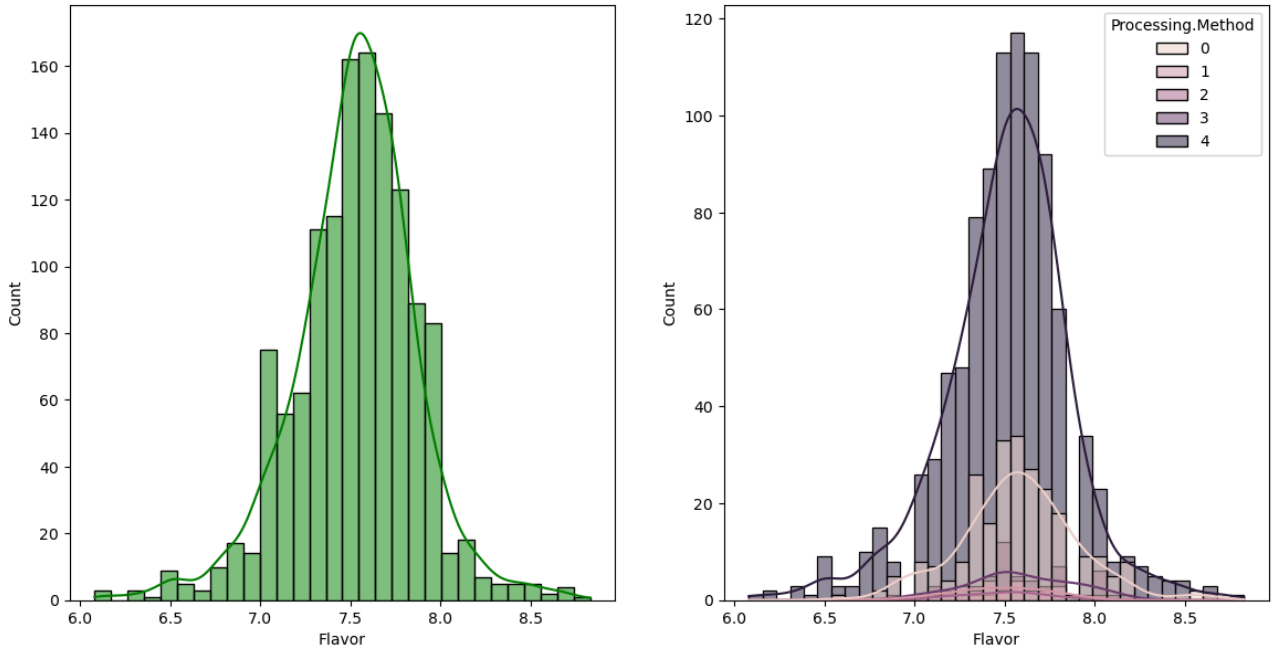
In [1085]:
```
# Correlation matrix of quality measures
correlation_matrix = quality_measures.corr()
plt.figure(figsize=(14, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Correlation Matrix of Quality Measure (Numerical Columns)')
plt.show()
```



In [1091]:
```
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
plt.subplot(121)
sns.histplot(data=df2,x='Flavor',bins=30,kde =True,color='g')
plt.subplot(122)
sns.histplot(data=df2,x='Flavor',kde =True,hue='Processing.Method')
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.

C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```

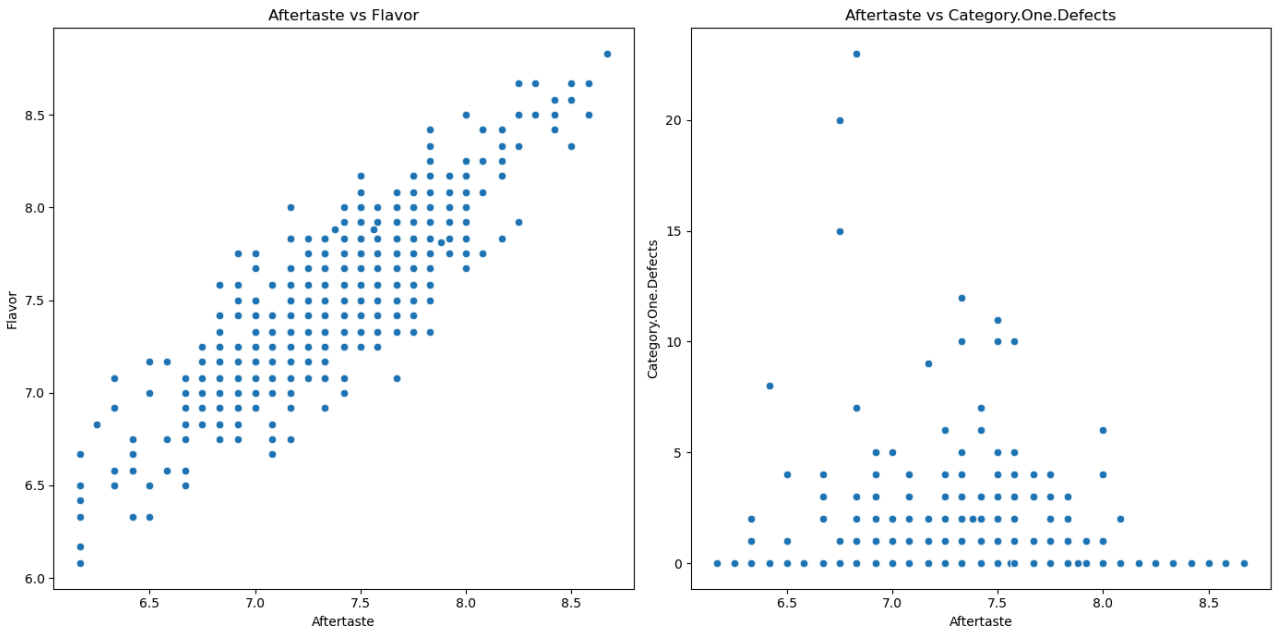
Out [1091]: <Axes: xlabel='Flavor', ylabel='Count'>



Comment : From the above graph the Washed/Wet method is used to process coffee beans more than other methods.

Scatter Plots for Aftertaste by some Quality Measures

In [1096]:
```
# Scatterplot example: [Aftertaste vs Flavor] and [Aftertaste vs Moisture]
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# ScatterPlot for Aftertaste vs Flavor :
sns.scatterplot(x='Aftertaste', y='Flavor', data=df2, ax=axs[0])
axs[0].set_title('Aftertaste vs Flavor')
# ScatterPlot for Aftertaste vs Category.One.Defects :
sns.scatterplot(x='Aftertaste', y='Category.One.Defects', data=df2, ax=axs[1])
axs[1].set_title('Aftertaste vs Category.One.Defects')
# Display the charts
plt.tight_layout()
plt.show()
```



Results :

- **Positive & Negative Correlation:** The data points form a clear upward trend, indicating a strong positive correlation between "Aftertaste" and "Flavor" whereareas the Aftertaste vs Category.One.Defects is negatively correlated.
- **This means that as the aftertaste score increases**, the flavor score also tends to increase whereaeas the opposite effect on Aftertaste and Category.One.Defects
- **Clustered Points:** Most data points are clustered between aftertaste scores of around 6 to 8 and flavor scores of around 6 to 8. This suggests that the majority of the samples have high scores in both aftertaste and flavor.

➽ Chi-Square Test (Chi2):

In [1107]:
```
df[categorical_columns].head() # 📣 after performing Chi-Square
```

Out [1107]:

	Species	Country.of.Origin	In.Country.Partner	Variety	Processing.Method	Color
0	0	8	15	5	4	2
1	0	8	15	15	4	2
2	0	9	19	2	4	2
3	0	8	15	5	0	2
4	0	8	15	15	4	2

In [1109]:

```
# Show results
chi2_results
```

Out [1109]:

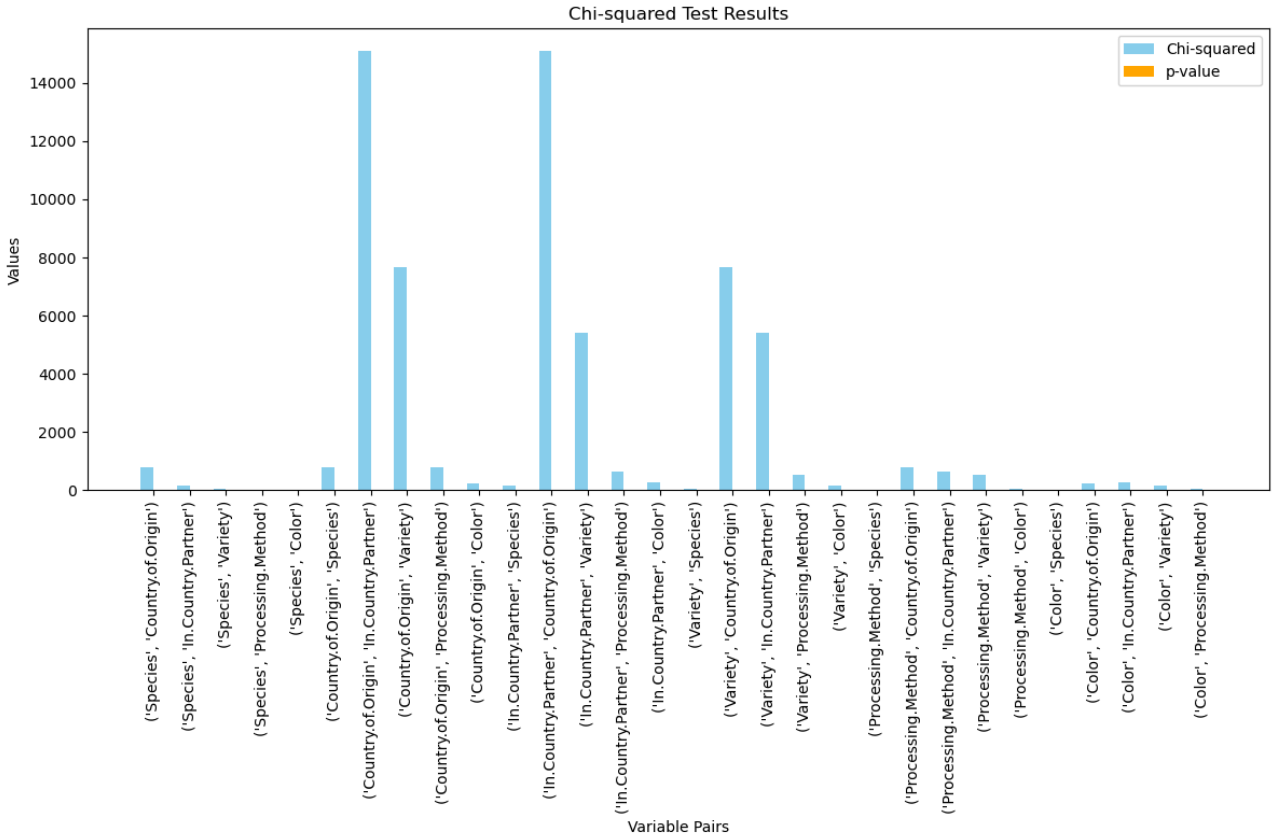
		chi2	p-value
Variable 1	**Variable 2**		
Species	**Country.of.Origin**	799.089834	3.099531e-145
	In.Country.Partner	178.878937	3.328648e-25
	Variety	43.084575	3.415564e-02
	Processing.Method	2.232002	6.931748e-01
	Color	0.990779	6.093336e-01
Country.of.Origin	**Species**	799.089834	3.099531e-145
	In.Country.Partner	15106.812537	0.000000e+00
	Variety	7651.534612	0.000000e+00
	Processing.Method	793.845899	6.006653e-92
	Color	226.846871	1.918919e-18
In.Country.Partner	**Species**	178.878937	3.328648e-25
	Country.of.Origin	15106.812537	0.000000e+00
	Variety	5396.600990	0.000000e+00
	Processing.Method	638.597458	2.108365e-79
	Color	267.143252	1.996102e-31
Variety	**Species**	43.084575	3.415564e-02
	Country.of.Origin	7651.534612	0.000000e+00
	In.Country.Partner	5396.600990	0.000000e+00
	Processing.Method	537.639406	7.390320e-57
	Color	156.941544	1.658389e-11
Processing.Method	**Species**	2.232002	6.931748e-01
	Country.of.Origin	793.845899	6.006653e-92
	In.Country.Partner	638.597458	2.108365e-79
	Variety	537.639406	7.390320e-57
	Color	44.417880	4.743247e-07
Color	**Species**	0.990779	6.093336e-01
	Country.of.Origin	226.846871	1.918919e-18
	In.Country.Partner	267.143252	1.996102e-31
	Variety	156.941544	1.658389e-11
	Processing.Method	44.417880	4.743247e-07

Charts for Chi-Square Test :

In [1113]:

```
# Extract data for plotting
pairs = chi2_results.index
chi2_values = chi2_results['chi2']
p_values = chi2_results['p-value']

# Set the positions for the bars
positions = range(len(pairs))

# Plotting the grouped bar chart
plt.figure(figsize=(12, 8))
bar_width = 0.35
plt.bar(positions, chi2_values, bar_width, label='Chi-squared', color='skyblue')
plt.bar([p + bar_width for p in positions], p_values, bar_width, label='p-value', color='orange')

# Add labels and title
plt.xlabel('Variable Pairs')
plt.ylabel('Values')
plt.title('Chi-squared Test Results')
plt.xticks([p + bar_width/2 for p in positions], pairs, rotation=90)
plt.legend()
```

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



In [1115]:
```
chi2_results.index = pd.MultiIndex.from_tuples(chi2_results.index, names=['Variable 1', 'Variable 2'])

# Create a pivot table for the heatmap
chi2_pivot = chi2_results.reset_index().pivot(index='Variable 1', columns='Variable 2', values='p-value')

# Create a heatmap to visualize the p-values
plt.figure(figsize=(14, 10))
sns.heatmap(chi2_pivot, annot=True, cmap='viridis', fmt=".4f", cbar_kws={'label': 'p-value'})
plt.title('Heatmap of Chi-Square Test p-values')
plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\matrix.py:260: FutureWarning:

Format strings passed to MaskedConstant are ignored, but in future may error or produce different behavior
```

Key Insights from Chi-Square Test:

✨ **Significant Associations:**

- **Species and Country of Origin**: The chi-squared statistic is 799.09 with a very low p-value (3.099531e-145), indicating a very strong association between coffee species and country of origin.
- **In Country Partner and Country of Origin**: The chi-squared statistic is 15106.81 with a p-value close to zero, indicating a strong association between in-country partner and country of origin.
- **Variety and Country of Origin**: The chi-squared statistic is 7651.53 with a p-value of zero, indicating a strong association between coffee variety and country of origin.
- **Processing Method and Country of Origin**: The chi-squared statistic is 793.85 with a p-value (6.006653e-92), showing a significant association between processing method and country of origin.
- **Color and Country of Origin**: The chi-squared statistic is 226.85 with a p-value (1.918919e-18), indicating a significant association between coffee color and country of origin.

✨ **Moderate Associations:**

- **Species and In Country Partner**: The chi-squared statistic is 178.88 with a p-value (3.328648e-25), suggesting a significant, though not as strong, association between species and in-country partner.
- **Processing Method and In Country Partner**: The chi-squared statistic is 638.60 with a p-value (2.108365e-79), indicating a significant association.
- **Variety and In Country Partner**: The chi-squared statistic is 5396.60 with a p-value of zero, showing a strong association.

✨ **Weak or No Associations:**

- **Processing Method and Species**: The chi-squared statistic is 2.23 with a high p-value (6.931748e-01), suggesting no significant association.
- **Color and Species**: The chi-squared statistic is 0.99 with a p-value (6.093336e-01), indicating no significant association.
- **Color and Processing Method**: The chi-squared statistic is 44.42 with a p-value (4.743247e-07), which is significant but not as strong compared to other associations.

🚀 **Analysed Relationships/ Intrepretation:**

- **Strongest Associations**: The Country of Origin appears to be a key factor influencing various other attributes of coffee, such as species, in-country partner, variety, processing method, and color.
- **Moderate Associations**: Attributes like species and in-country partner have notable associations, which might be relevant for understanding regional coffee characteristics.
- **Weak Associations**: There is little to no significant relationship between processing method and species, as well as color and species, suggesting these attributes may vary independently of each other.

➽ ANOVA (F_classif):

ANOVA Feature Classif : (NUM to CAT)

- *Selecting Total.Cup.Points as target (y) beacause it seems like a strong candidate for the target variable as it directly indicates the overall quality of the coffee.*

- *And also comparing some other numerical features with categorical features.*

In [1130]:
```
display(Markdown(bold_text12))
print("Selected Features for Total.Cup.Points:", selected_features)
print("Selected Features for Flavor:", selected_features)
print("Selected Features for Balance:", selected_features)
print("Selected Features of Acidity:", selected_features)
```

```
<IPython.core.display.Markdown object>

Selected Features for Total.Cup.Points: Index(['Body', 'Quakers'], dtype='object')
Selected Features for Flavor: Index(['Body', 'Quakers'], dtype='object')
Selected Features for Balance: Index(['Body', 'Quakers'], dtype='object')
Selected Features of Acidity: Index(['Body', 'Quakers'], dtype='object')
```

Insights from Selected Features for Coffee Quality Measures

Total.Cup.Points:

Selected Features:

- `In.Country.Partner`
- `Processing.Method`

Insights:

- **In.Country.Partner**: The relationship with the in-country partner, likely referring to the local entity or organization involved in the coffee's production or quality assessment, plays a significant role in the overall cup score. This could be due to the varying standards and practices of different partners in processing and evaluating coffee.
- **Processing.Method**: The method used to process the coffee beans (e.g., washed, natural, honey) has a significant impact on the total cup points. This is expected as the processing method affects the flavor profile, cleanliness, and other sensory attributes of the coffee.

Flavor:

Selected Features:

- `Country.of.Origin`
- `In.Country.Partner`

Insights:

- **Country.of.Origin**: The origin country of the coffee is a critical determinant of its flavor. Different countries, due to their unique climates, soil types, and coffee varieties, produce beans with distinctive flavor profiles.
- **In.Country.Partner**: The local partner's influence also plays a significant role in the flavor, suggesting that local practices and expertise in handling and processing coffee beans can significantly alter the flavor profile.

Balance:

Selected Features:

- `Species`
- `In.Country.Partner`

Insights:

- **Species**: The coffee species (e.g., Arabica, Robusta) is crucial in determining the balance of the coffee. Different species have inherent characteristics that influence how balanced the flavor profile is.
- **In.Country.Partner**: Again, the involvement of the local partner is significant, indicating that their practices in growing, harvesting, and processing the beans can affect the balance of the coffee.

Acidity:

Selected Features:

- `In.Country.Partner`
- `Variety`

Insights:

- **In.Country.Partner**: The local partner's impact on acidity is noteworthy, suggesting that local methods and environmental factors managed by the partner affect the acidity levels in coffee.
- **Variety**: The specific variety of coffee (e.g., Bourbon, Typica) influences the acidity. Different varieties have genetic differences that result in varying acidity levels in the cup.

General Insights:

- **In.Country.Partner**: This feature appears consistently across all quality measures, highlighting the significant role of local partners in influencing coffee quality. This could encompass their expertise, quality control measures, and specific practices in coffee handling and processing.
- **Processing.Method**: Specifically influential for the total cup points, indicating the critical impact of how the coffee is processed after harvest.
- **Country.of.Origin**: Plays a crucial role in flavor, emphasizing the importance of geographical and climatic conditions in shaping the sensory characteristics of coffee.
- **Species and Variety**: These genetic factors are essential for specific attributes like balance and acidity, underscoring the intrinsic qualities of the coffee plants themselves.

These insights can guide stakeholders in the coffee industry, such as farmers, processors, and marketers, to focus on these critical factors to enhance coffee quality. Understanding these relationships can help in making informed decisions about cultivation practices, processing methods, and partnership selections to achieve desired quality outcomes.

AUTO ANOVA Feature Classif : (CAT to NUM)

- *Now Comparing the Categorical Features with Numerical Features of 'Country.of.Origin' and 'Processing.Method'.*

In [1136]:
```
print(" Country.of.Origin with Numerical Features of :")
anova_results_cat
```

Country.of.Origin with Numerical Features of :

Out [1136]:

	Feature	F Stat	P-value
0	Number.of.Bags	25.480386	1.764103e-121
1	Bag.Weight	1.523371	2.662345e-02
2	Aroma	5.272383	1.899711e-20
3	Flavor	9.046545	3.158334e-41
4	Aftertaste	11.667931	2.461880e-55
5	Acidity	10.183243	2.131956e-47
6	Body	11.667481	2.475422e-55
7	Balance	11.983125	5.317317e-57
8	Uniformity	3.104323	5.206005e-09
9	Clean.Cup	2.417370	9.029156e-06
10	Sweetness	16.249817	8.014149e-79
11	Cupper.Points	7.766437	3.352928e-34
12	Total.Cup.Points	8.774658	9.678481e-40
13	Moisture	11.415124	5.398175e-54
14	Category.One.Defects	2.628322	9.938988e-07
15	Quakers	1.090843	3.300645e-01
16	Category.Two.Defects	5.590570	3.421925e-22

In [1139]:
```
print(" Processing.Method with Numerical Features of :")
anova_results_cat1
```

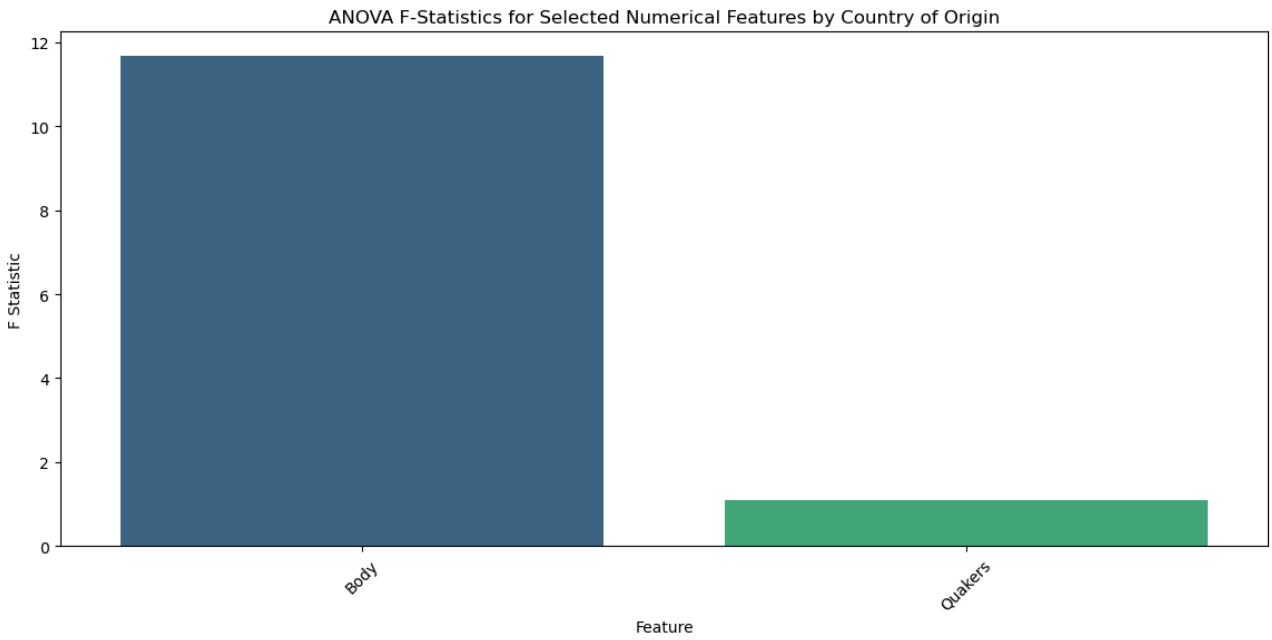
Processing.Method with Numerical Features of :

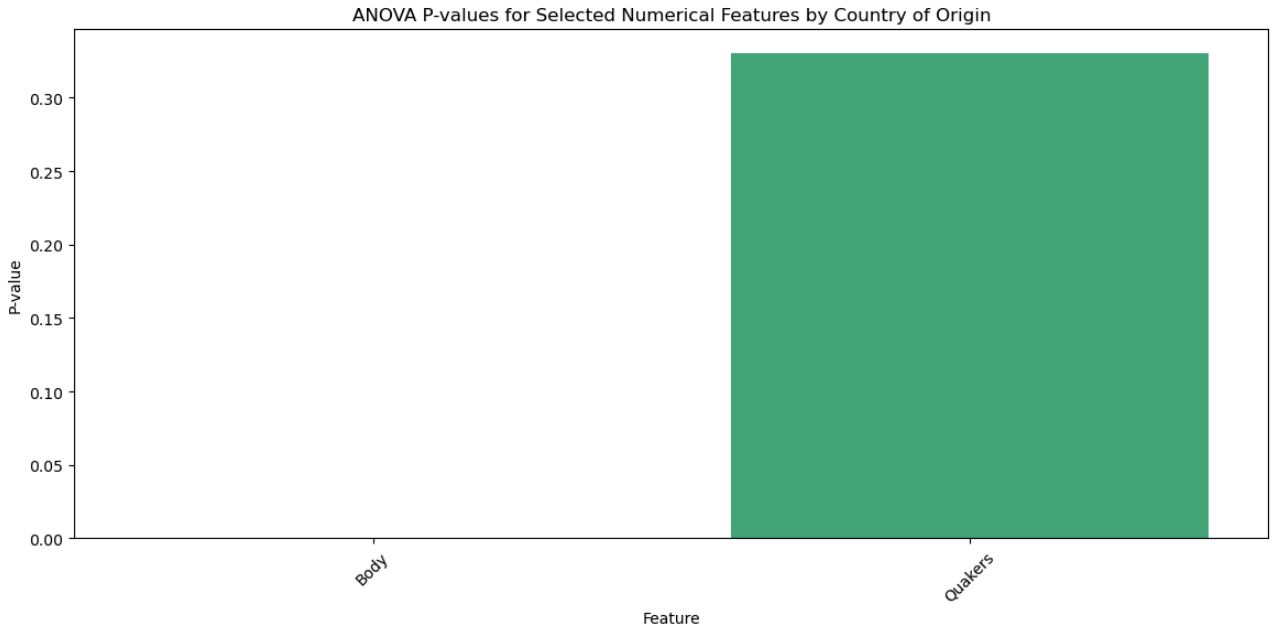
Out [1139]:

	Feature	F Stat	P-value
0	Number.of.Bags	3.374498	9.318407e-03
1	Bag.Weight	0.152869	9.617434e-01
2	Aroma	1.023394	3.939225e-01
3	Flavor	2.837926	2.327918e-02
4	Aftertaste	2.772290	2.599877e-02
5	Acidity	1.264402	2.820331e-01
6	Body	5.429583	2.455748e-04
7	Balance	2.995515	1.782911e-02
8	Uniformity	1.243548	2.905402e-01
9	Clean.Cup	0.815089	5.154887e-01
10	Sweetness	4.340513	1.719830e-03
11	Cupper.Points	3.395521	8.986515e-03
12	Total.Cup.Points	1.648836	1.595869e-01
13	Moisture	3.080156	1.543726e-02
14	Category.One.Defects	2.273576	5.937978e-02
15	Quakers	15.396516	2.601653e-12
16	Category.Two.Defects	0.493770	7.403395e-01

Charts for ANOVA F-Test :

In [1144]:
```
# Visualize the selected features' ANOVA F-statistics and p-values
selected_anova_results = anova_results_cat[anova_results_cat['Feature'].isin(selected_features)]

# Plot F-statistics
plt.figure(figsize=(14, 6))
sns.barplot(x='Feature', y='F Stat', data=selected_anova_results, palette='viridis')
plt.title('ANOVA F-Statistics for Selected Numerical Features by Country of Origin')
plt.xlabel('Feature')
plt.ylabel('F Statistic')
plt.xticks(rotation=45)
plt.show()

# Plot p-values
plt.figure(figsize=(14, 6))
sns.barplot(x='Feature', y='P-value', data=selected_anova_results, palette='viridis')
plt.title('ANOVA P-values for Selected Numerical Features by Country of Origin')
plt.xlabel('Feature')
plt.ylabel('P-value')
plt.xticks(rotation=45)
plt.show()
```





General Insights :

Selected Features for Coffee Quality Measures

- **Number.of.Bags & Sweetness**:
  - **Number.of.Bags**: Highly significant (F Stat: 25.480386, P-value: 1.764103e-121) influencing quality measures.
  - **Sweetness**: Very significant (F Stat: 16.249817, P-value: 8.014149e-79) affecting various quality attributes.

Top ANOVA Results for Coffee Quality

- **Balance**: Highly significant (F Stat: 11.983125, P-value: 5.317317e-57).
- **Aftertaste**: Strong impact (F Stat: 11.667931, P-value: 2.461880e-55).
- **Body**: Significant (F Stat: 11.667481, P-value: 2.475422e-55).
- **Sweetness**: Notable influence (F Stat: 16.249817, P-value: 8.014149e-79).

Selected Features for Processing Method

- **Body & Quakers**:
  - **Body**: Significant influence (F Stat: 5.429583, P-value: 2.455748e-04).
  - **Quakers**: Highly significant (F Stat: 15.396516, P-value: 2.601653e-12).

Summary

- **Key Influencers**: Number of Bags, Sweetness, Balance, Aftertaste, and Body are major factors affecting coffee quality.
- **Processing Method**: Body and Quakers are crucial for determining the processing method's impact.

==========

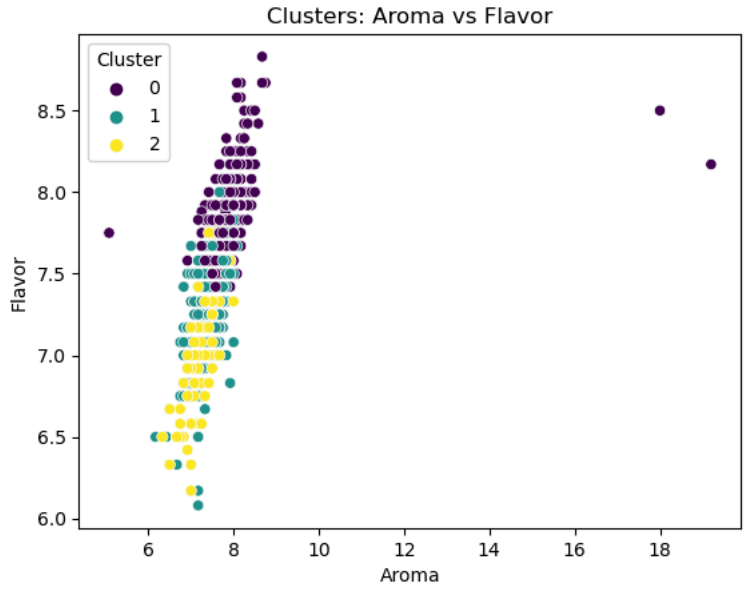
12) Multivariate Analysis :

- *Exploration of complex relationships involving multiple variables.*

➽ Clustering:

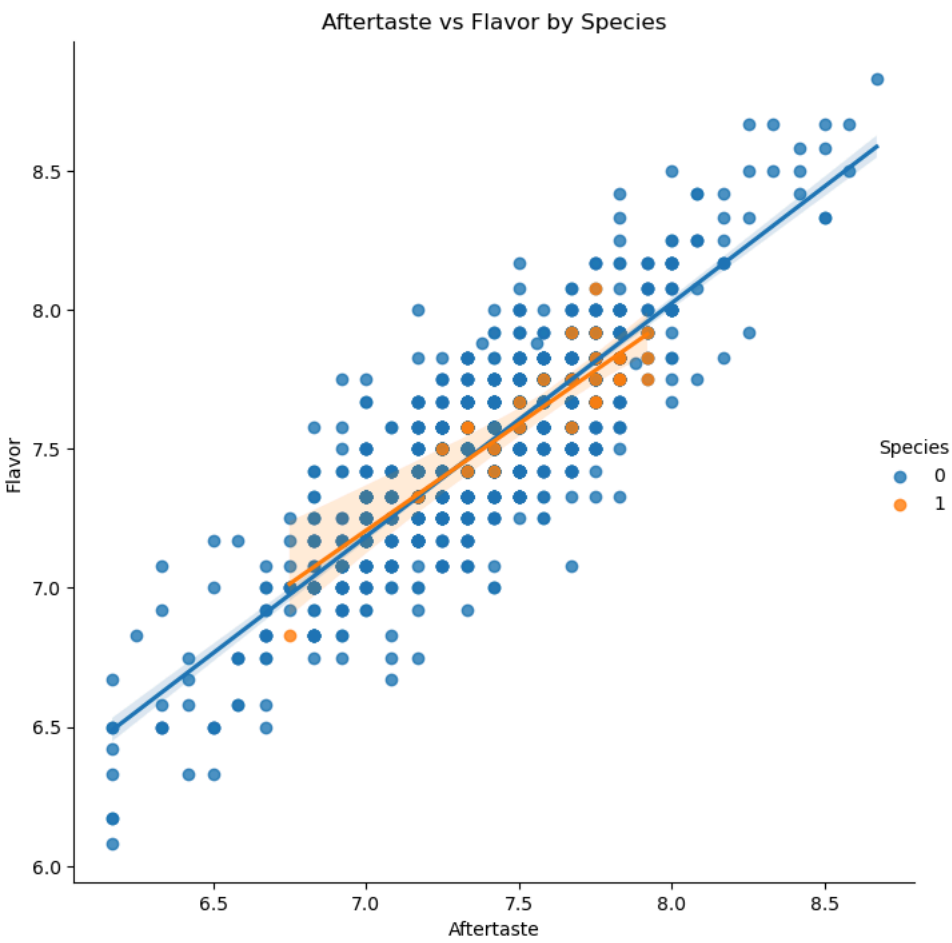
In [1151]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(quality_measures)
# KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df2['Cluster'] = kmeans.fit_predict(data_scaled)
# Plot clusters
plt.figure()
sns.scatterplot(x='Aroma', y='Flavor', hue='Cluster', data=df2, palette='viridis')
plt.title('Clusters: Aroma vs Flavor')
plt.show()
```

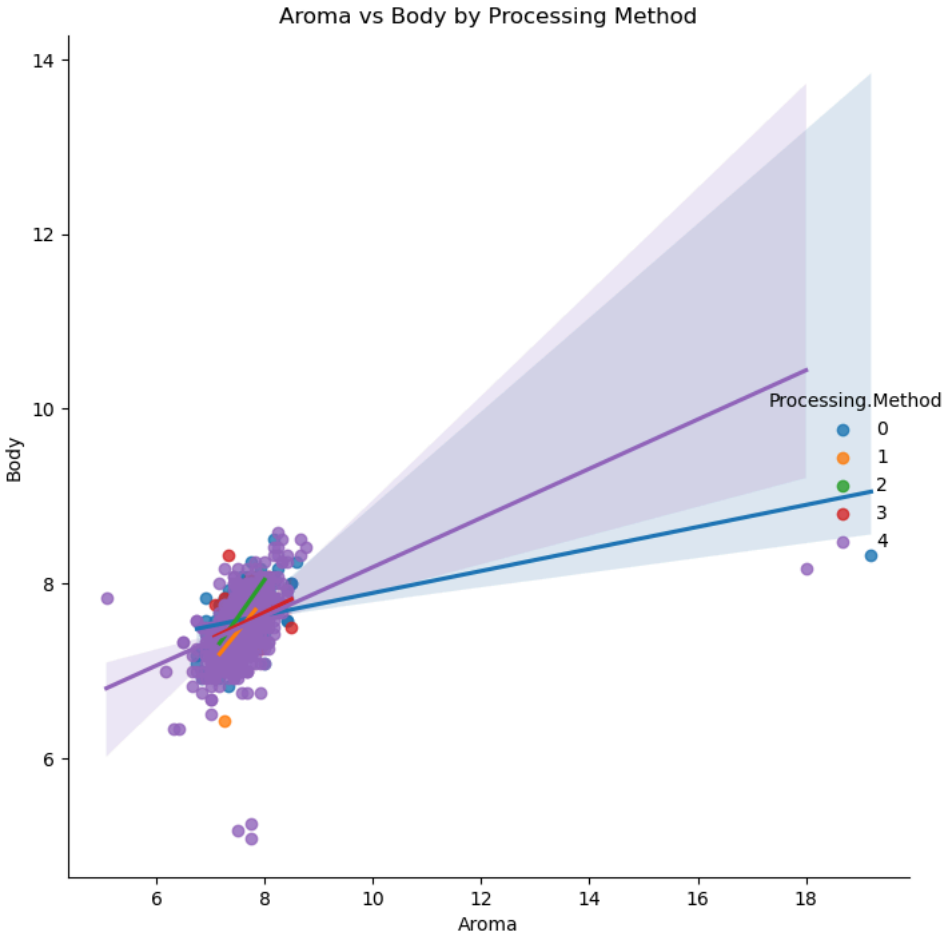
```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```



Some other comparison in lmplot :

In [1155]:
```
# Create the first lmplot and adjust its position
g1 = sns.lmplot(x='Aftertaste', y='Flavor', hue='Species', data=df2, aspect=1.5)
g1.set_axis_labels('Aftertaste', 'Flavor')
g1.fig.set_size_inches(7, 7)
g1.fig.subplots_adjust(left=0.05, right=0.95, top=0.95, bottom=0.05)
g1.fig.suptitle('Aftertaste vs Flavor by Species')

# Create the second lmplot and adjust its position
g2 = sns.lmplot(x='Aroma', y='Body', hue='Processing.Method', data=df2, aspect=1.5)
g2.set_axis_labels('Aroma', 'Body')
g2.fig.set_size_inches(7, 7)
g2.fig.subplots_adjust(left=0.05, right=0.95, top=0.95, bottom=0.05)
g2.fig.suptitle('Aroma vs Body by Processing Method')

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



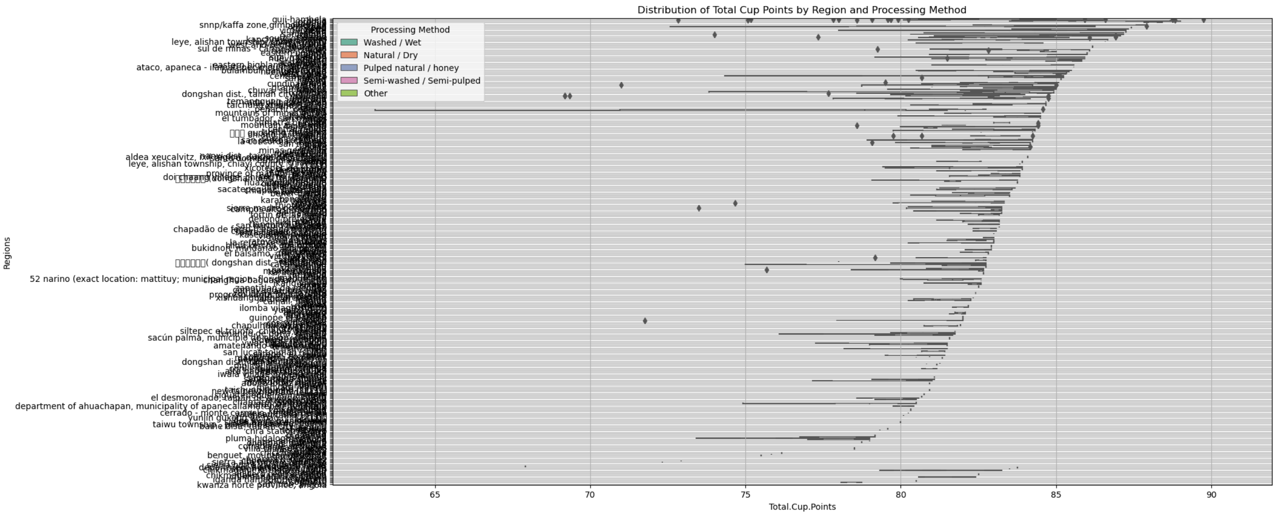


In [1439]:

```
# Create a grouped box plot using Seaborn
plt.figure(figsize=(20, 10))
sns.boxplot(x='Total.Cup.Points', y='Region', hue='Processing.Method', data=df4, palette='Set2')
plt.title('Distribution of Total Cup Points by Region and Processing Method')
plt.xlabel('Total.Cup.Points')
plt.ylabel('Regions')
plt.legend(title='Processing Method', loc='upper left')
plt.grid(True)
plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 33274 (\N{CJK UNIFIED IDEOGRAPH-81FA}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21335 (\N{CJK UNIFIED IDEOGRAPH-5357}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 24066 (\N{CJK UNIFIED IDEOGRAPH-5E02}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 26481 (\N{CJK UNIFIED IDEOGRAPH-6771}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 23665 (\N{CJK UNIFIED IDEOGRAPH-5C71}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21312 (\N{CJK UNIFIED IDEOGRAPH-5340}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21488 (\N{CJK UNIFIED IDEOGRAPH-53F0}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 20013 (\N{CJK UNIFIED IDEOGRAPH-4E2D}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 26032 (\N{CJK UNIFIED IDEOGRAPH-65B0}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 31038 (\N{CJK UNIFIED IDEOGRAPH-793E}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 22283 (\N{CJK UNIFIED IDEOGRAPH-570B}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 22995 (\N{CJK UNIFIED IDEOGRAPH-59D3}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 37129 (\N{CJK UNIFIED IDEOGRAPH-9109}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 26976 (\N{CJK UNIFIED IDEOGRAPH-6960}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 35199 (\N{CJK UNIFIED IDEOGRAPH-897F}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 22025 (\N{CJK UNIFIED IDEOGRAPH-5609}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 32681 (\N{CJK UNIFIED IDEOGRAPH-7FA9}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 38463 (\N{CJK UNIFIED IDEOGRAPH-963F}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 37324 (\N{CJK UNIFIED IDEOGRAPH-91CC}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 27138 (\N{CJK UNIFIED IDEOGRAPH-6A02}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 37326 (\N{CJK UNIFIED IDEOGRAPH-91CE}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 26449 (\N{CJK UNIFIED IDEOGRAPH-6751}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 28771 (\N{CJK UNIFIED IDEOGRAPH-7063}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21644 (\N{CJK UNIFIED IDEOGRAPH-548C}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 24179 (\N{CJK UNIFIED IDEOGRAPH-5E73}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 25237 (\N{CJK UNIFIED IDEOGRAPH-6295}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 32291 (\N{CJK UNIFIED IDEOGRAPH-7E23}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 22826 (\N{CJK UNIFIED IDEOGRAPH-592A}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 40635 (\N{CJK UNIFIED IDEOGRAPH-9EBB}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 24432 (\N{CJK UNIFIED IDEOGRAPH-5F70}) missing from current font.
```

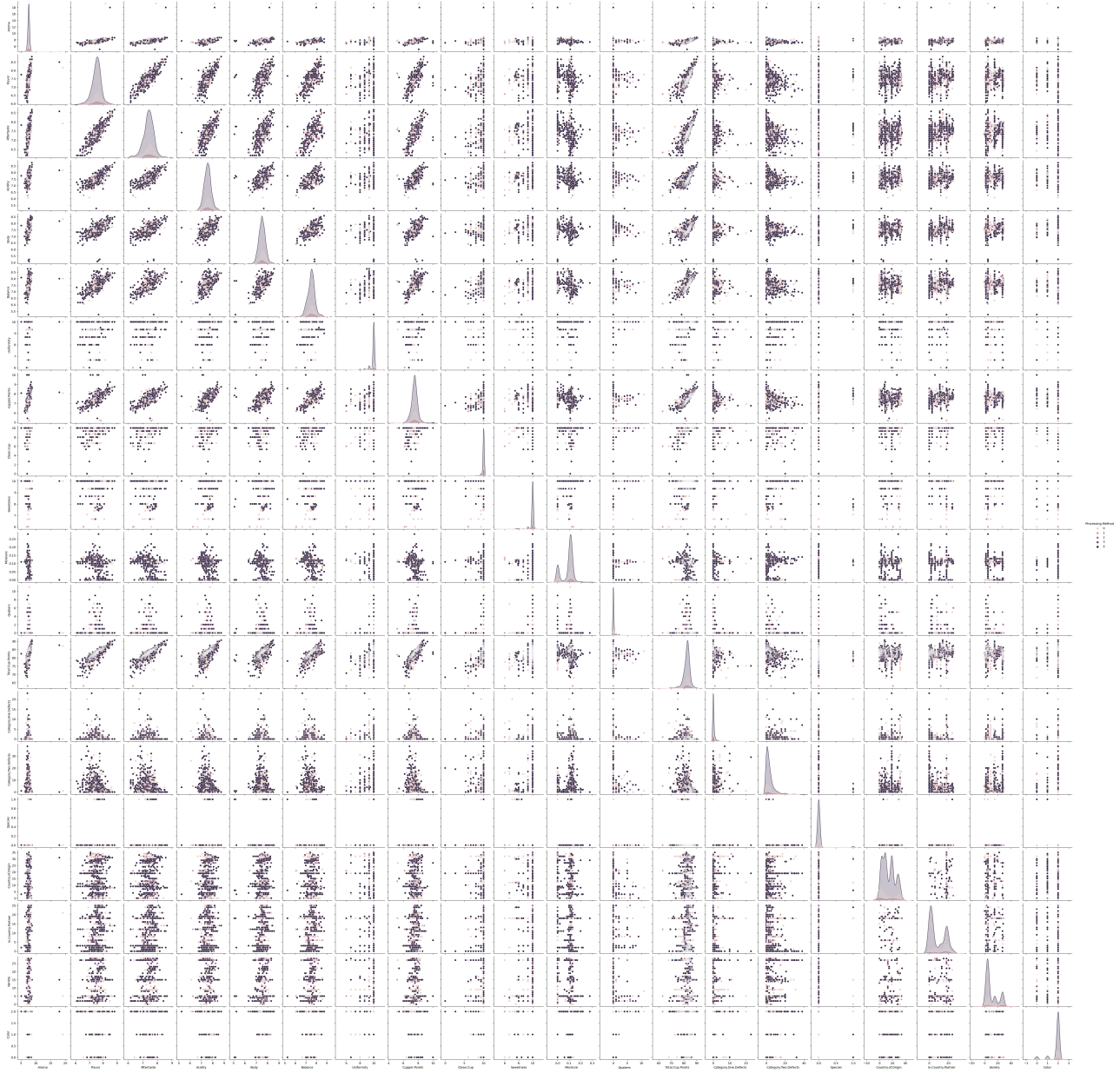
```
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21270 (\N{CJK UNIFIED IDEOGRAPH-5316}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 20843 (\N{CJK UNIFIED IDEOGRAPH-516B}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21350 (\N{CJK UNIFIED IDEOGRAPH-5366}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 38642 (\N{CJK UNIFIED IDEOGRAPH-96F2}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 26519 (\N{CJK UNIFIED IDEOGRAPH-6797}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 30707 (\N{CJK UNIFIED IDEOGRAPH-77F3}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 22721 (\N{CJK UNIFIED IDEOGRAPH-58C1}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21476 (\N{CJK UNIFIED IDEOGRAPH-53E4}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 22353 (\N{CJK UNIFIED IDEOGRAPH-5751}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 33655 (\N{CJK UNIFIED IDEOGRAPH-8377}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21253 (\N{CJK UNIFIED IDEOGRAPH-5305}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 23574 (\N{CJK UNIFIED IDEOGRAPH-5C16}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 34399 (\N{CJK UNIFIED IDEOGRAPH-865F}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 33495 (\N{CJK UNIFIED IDEOGRAPH-82D7}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 26647 (\N{CJK UNIFIED IDEOGRAPH-6817}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 19977 (\N{CJK UNIFIED IDEOGRAPH-4E09}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 21271 (\N{CJK UNIFIED IDEOGRAPH-5317}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 30058 (\N{CJK UNIFIED IDEOGRAPH-756A}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 36335 (\N{CJK UNIFIED IDEOGRAPH-8DEF}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 27888 (\N{CJK UNIFIED IDEOGRAPH-6CF0}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 23433 (\N{CJK UNIFIED IDEOGRAPH-5B89}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 33502 (\N{CJK UNIFIED IDEOGRAPH-82DE}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 23631 (\N{CJK UNIFIED IDEOGRAPH-5C4F}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 27494 (\N{CJK UNIFIED IDEOGRAPH-6B66}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 30333 (\N{CJK UNIFIED IDEOGRAPH-767D}) missing from current font.
C:\Users\Admin\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning:
Glyph 27827 (\N{CJK UNIFIED IDEOGRAPH-6CB3}) missing from current font.
```



Multivariate Analysis using Pairplot

In [1160]:
```
sns.pairplot(pairplot_chart,hue= 'Processing.Method')
plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```

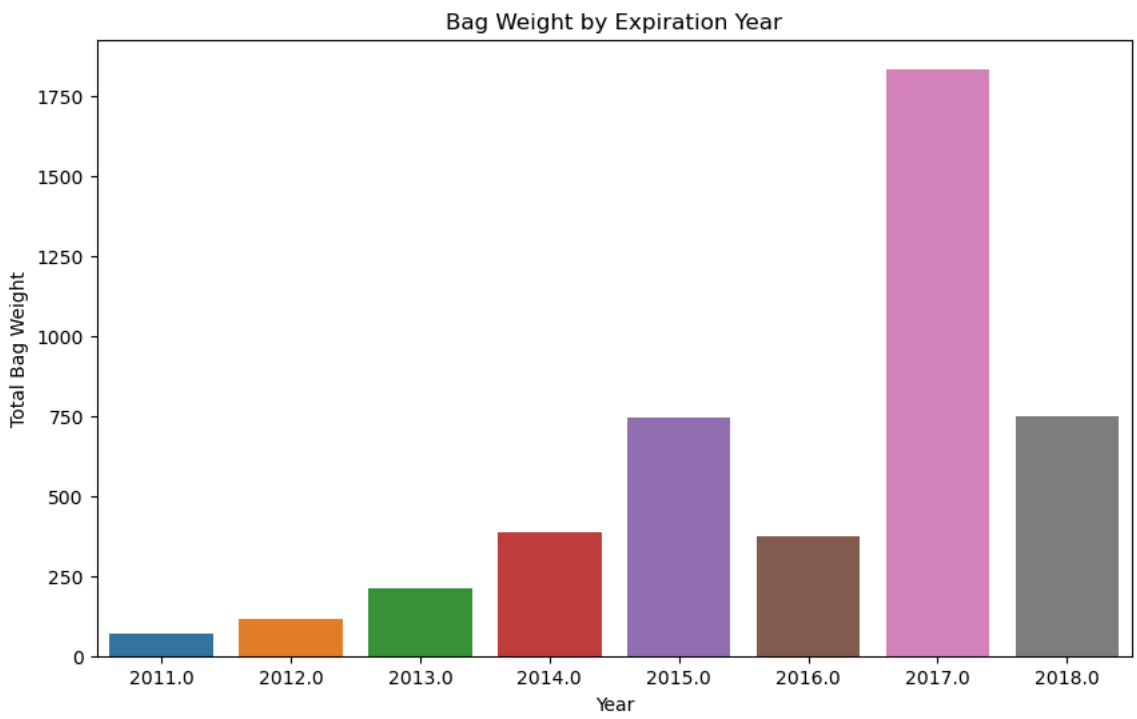


Let's get to see some charts before going to Overall Insights Obtained from Analysis

Combination Viz 1 : *Let's Combine "**Bag.Weight**" and "**Expiration Year**" for some visualization.*

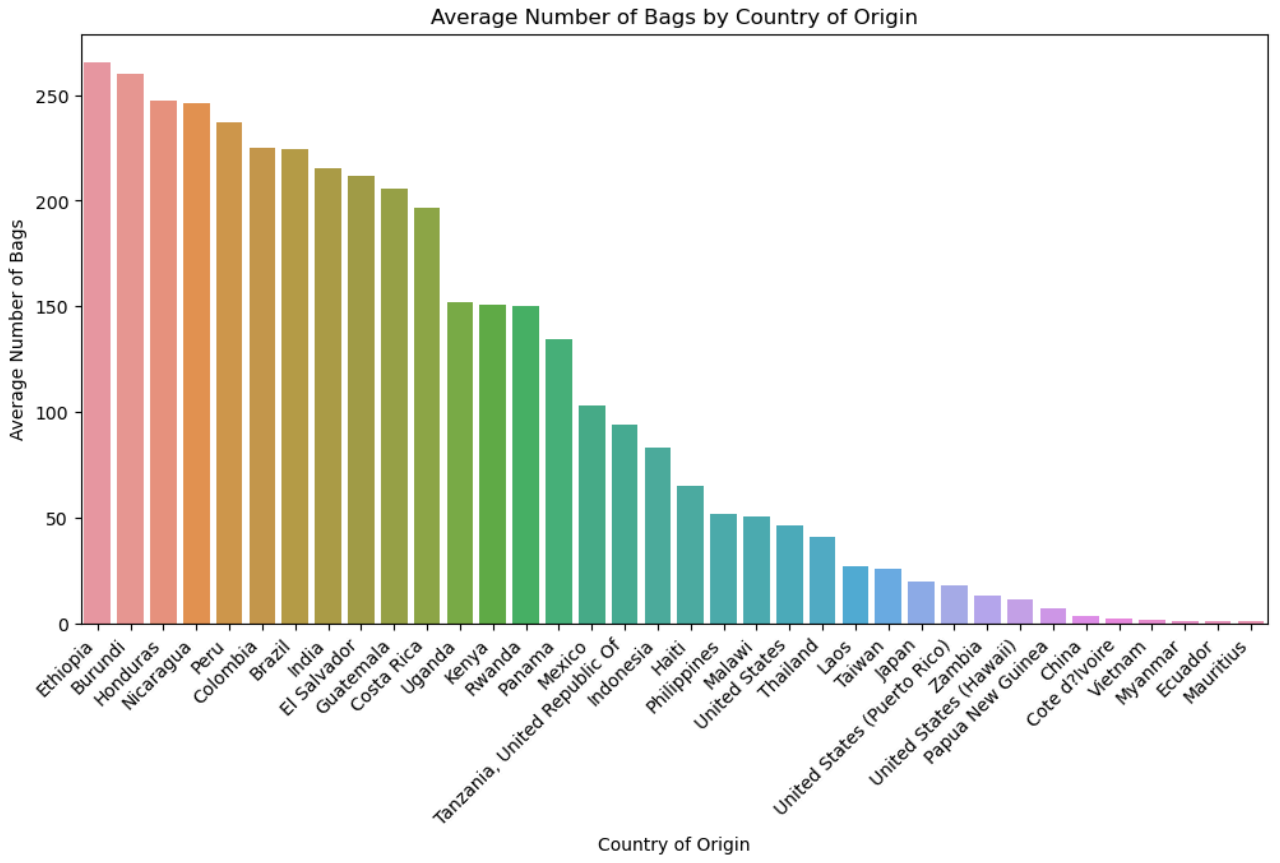
In [1308]:

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Year', y='Bag.Weight', data=df_yearly)
plt.title('Bag Weight by Expiration Year')
plt.xlabel('Year')
plt.ylabel('Total Bag Weight')
plt.show()
```

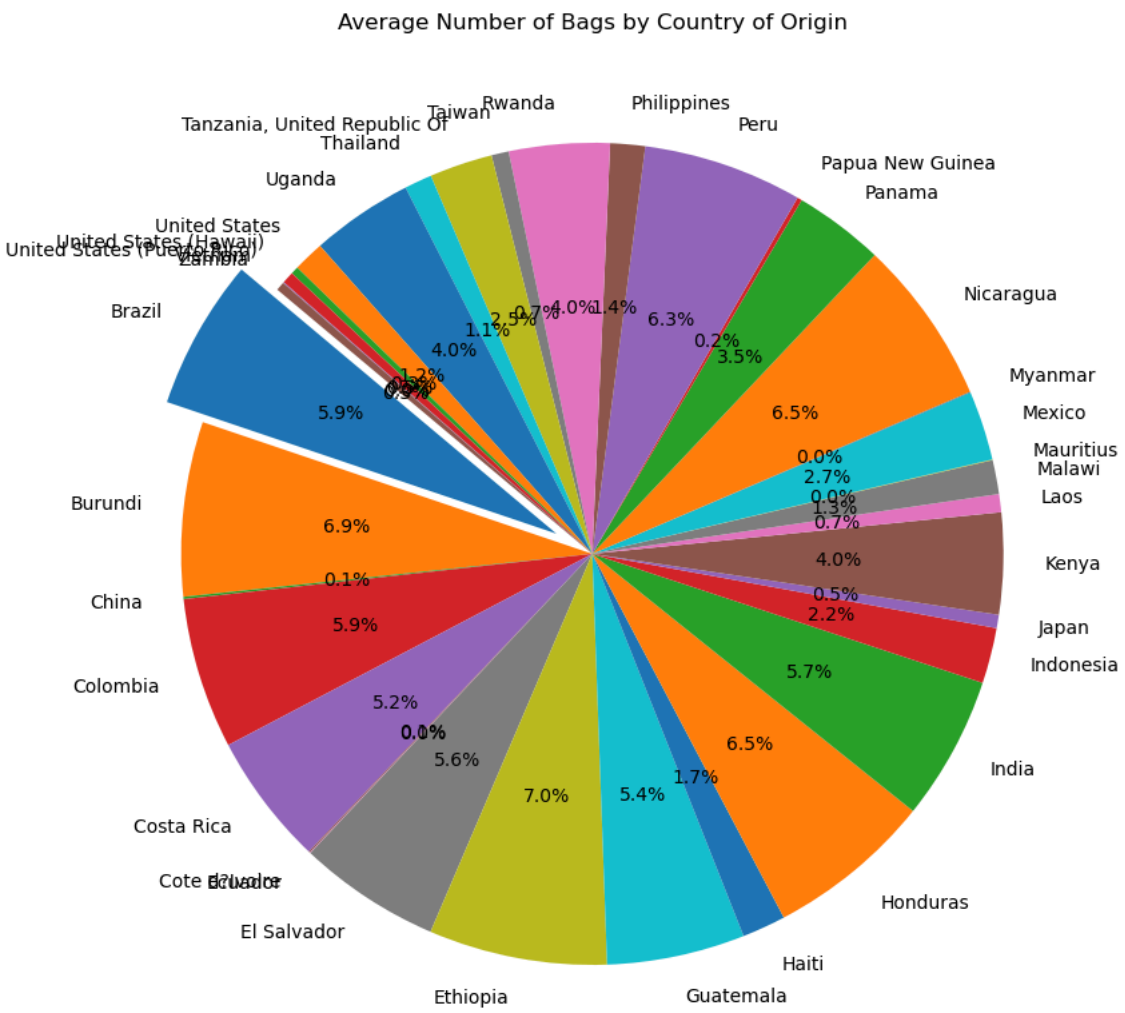


Combination Viz 2 : *Let's Combine "**Country.of.Origin**" and "**Number.of.Bags**" for some visualization.*

In [1310]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='Country.of.Origin', y='Number.of.Bags', data=average_bags_per_origin, order=sorted_countries, e
plt.title('Average Number of Bags by Country of Origin')
plt.xlabel('Country of Origin')
plt.ylabel('Average Number of Bags')
plt.xticks(rotation=45, ha='right')
plt.show()
```



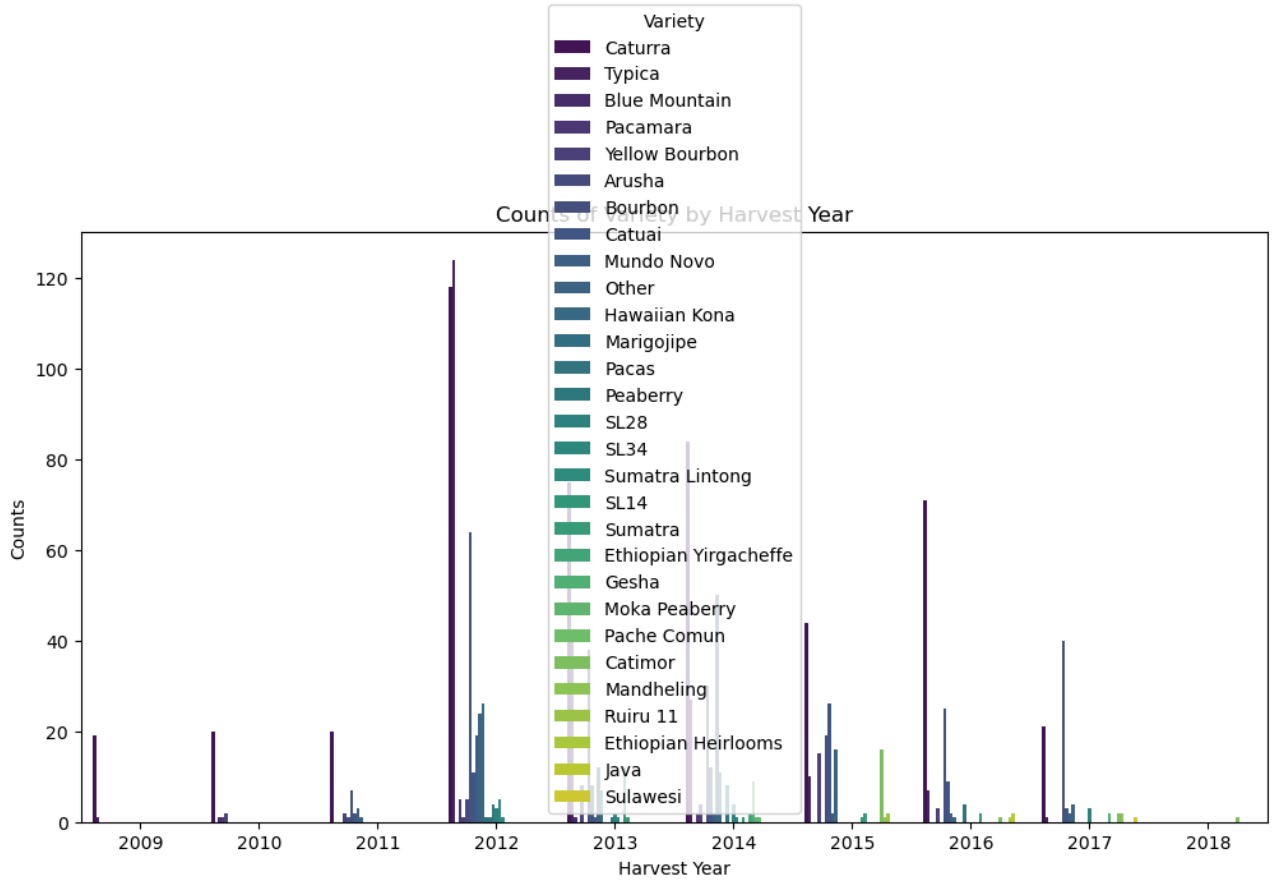
In [1312]:
```
sizes = average_bags_per_origin['Number.of.Bags']
labels = average_bags_per_origin['Country.of.Origin']
explode = [0.1 if i == 0 else 0 for i in range(len(sizes))]
plt.figure(figsize=(10, 10))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, explode=explode)
plt.title('Average Number of Bags by Country of Origin')
plt.show()
```



Combination Viz 3 : *Let's Combine "**Harvest.Year**" and "**Variety**" for some visualization.*

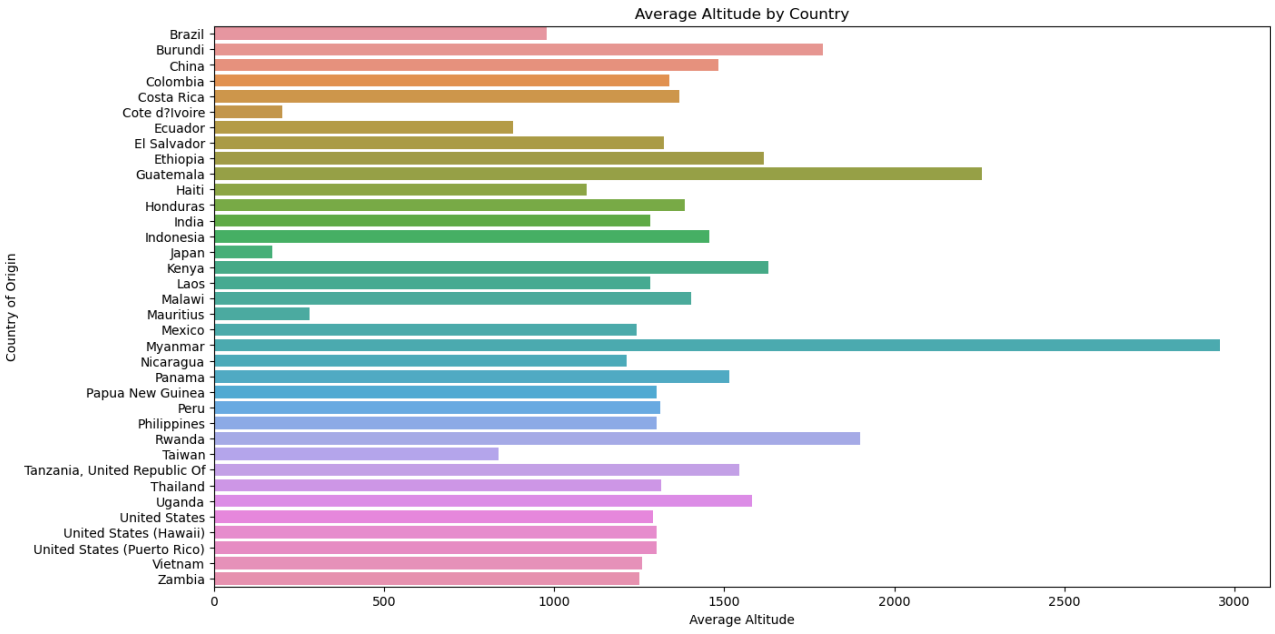
In [1315]:

```
plt.figure(figsize=(12,6))
sns.barplot(data=farm_year_counts, x='Harvest.Year', y='counts', hue='Variety', palette='viridis')
plt.xlabel('Harvest Year')
plt.ylabel('Counts')
plt.title('Counts of Variety by Harvest Year')
plt.show()
```

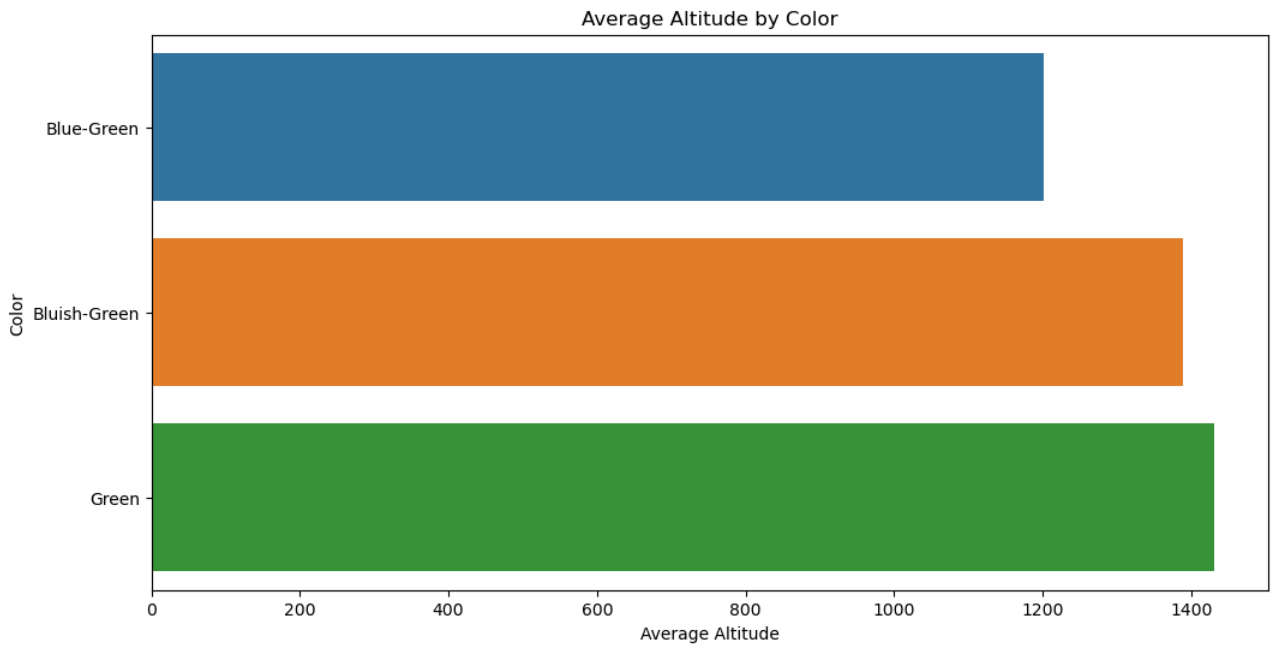


Combination Viz 4 : *Let's Combine columns with "**Altitude**" for some visualization and analysing*

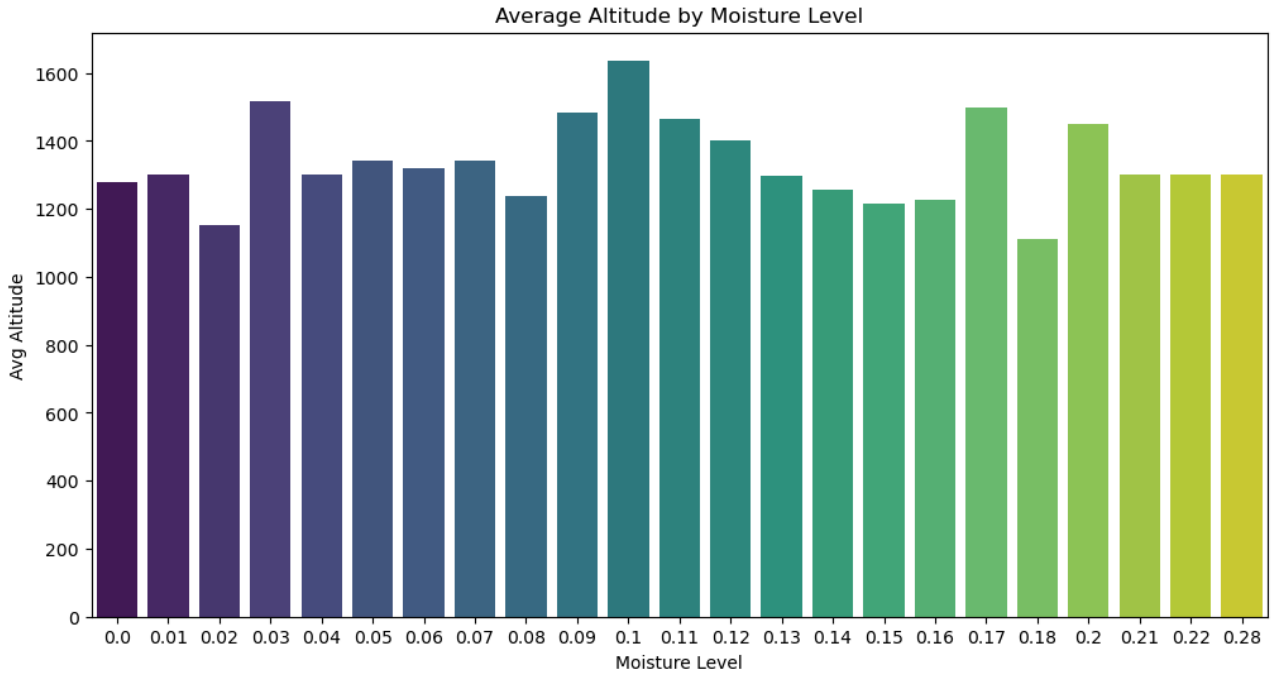
In [1317]:
```
plt.figure(figsize=(15, 8))
sns.barplot(x='Altitude', y='Country.of.Origin', data=avg_altitude_by_country)
plt.title('Average Altitude by Country')
plt.xlabel('Average Altitude')
plt.ylabel('Country of Origin')
plt.show()
```



In [1323]:
```
# Does altitude affect coffee color?
avg_altitude_by_color = df4.groupby('Color')['Altitude'].mean().reset_index()
plt.figure(figsize=(12, 6))
sns.barplot(x='Altitude', y='Color', data=avg_altitude_by_color)
plt.title('Average Altitude by Color')
plt.xlabel('Average Altitude')
plt.ylabel('Color')
plt.show()
```



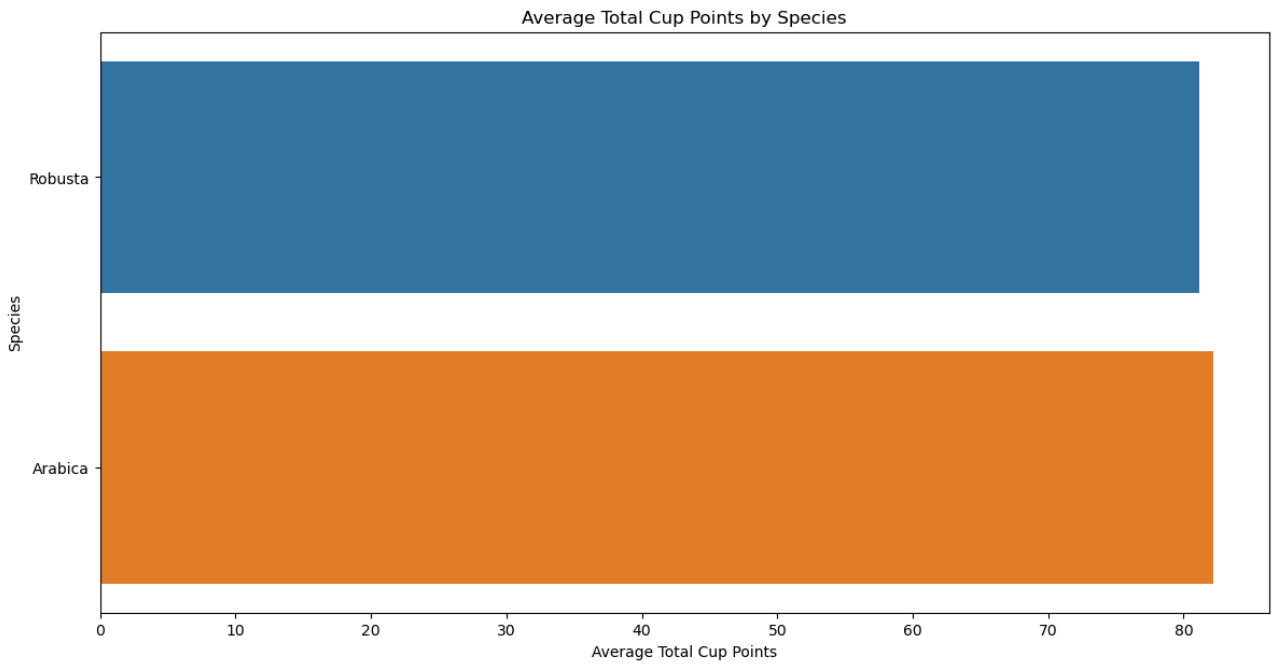
In [1325]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='Moisture', y='Altitude', data=avg_altitude_by_moisture, palette='viridis')
plt.title('Average Altitude by Moisture Level')
plt.xlabel('Moisture Level')
plt.ylabel('Avg Altitude')
plt.show()
```



Combination Viz 5 : *Let's Combine **"Species"** and **"Total.Cup.Points"** for some visualization.*

In [1391]:
```
Spect1
```

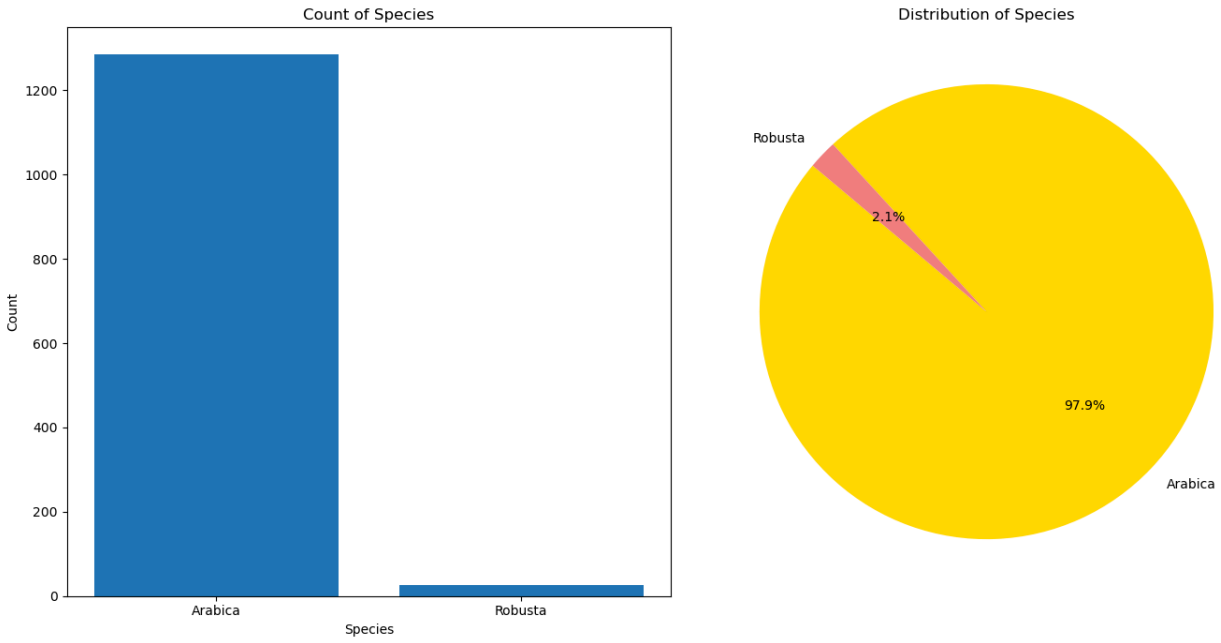
Out [1391]:



In [1384]:

```
Spect
```

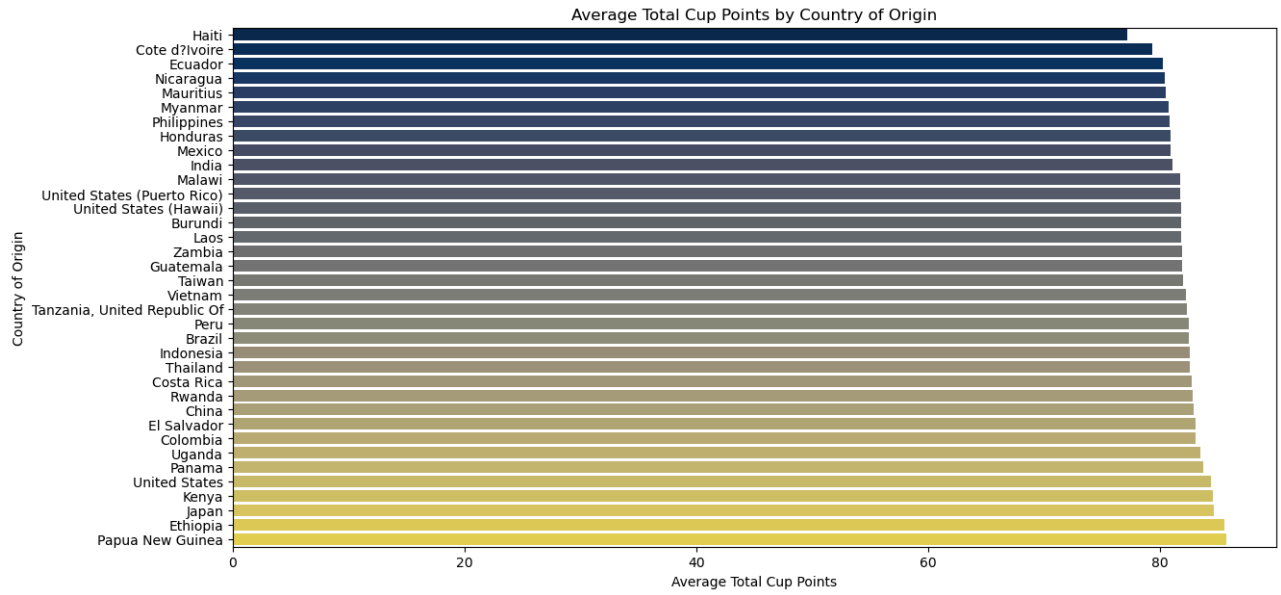
Out [1384]:



Combination Viz 6 : *Let's Combine "**Country.of.Origin**" and "**Total.Cup.Points**" for some visualization.*

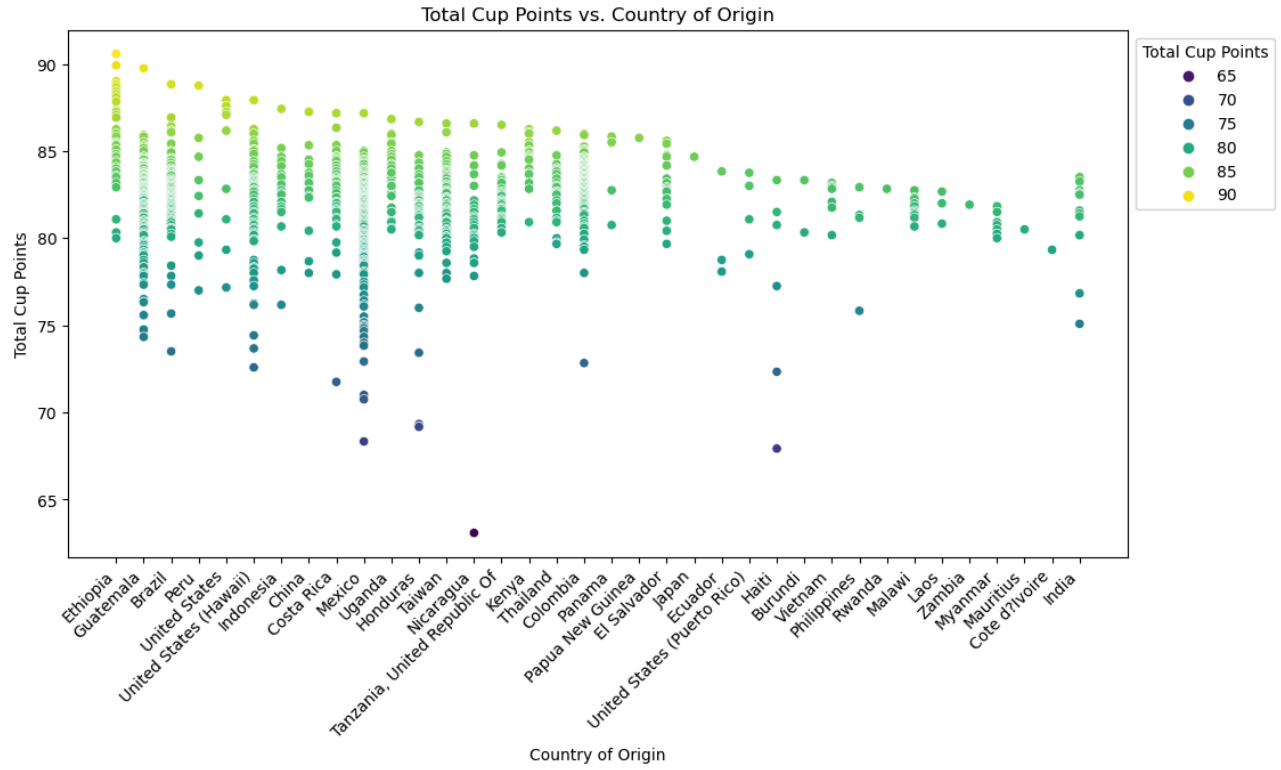
In [1343]:

```
plt.figure(figsize=(14,7))
sns.barplot(x='Total.Cup.Points', y='Country.of.Origin', data=avg_total_cup_points_by_country, order=avg_total_
plt.title('Average Total Cup Points by Country of Origin')
plt.xlabel('Average Total Cup Points')
plt.ylabel('Country of Origin')
plt.show()
```



In [1345]:

```
color_palette = sns.color_palette("viridis", as_cmap=True)
# Scatterplot
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Country.of.Origin', y='Total.Cup.Points', data=df4, hue='Total.Cup.Points', palette=color_pa
plt.title('Total Cup Points vs. Country of Origin')
plt.xlabel('Country of Origin')
plt.ylabel('Total Cup Points')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Total Cup Points', loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
```



In [1353]:

```
import plotly.express as px
# Average Total Cup Points by Country of Origin
fig = px.choropleth(data,
                    locations='Country.of.Origin',
                    locationmode='country names',
                    color='Total.Cup.Points',
                    hover_name='Country.of.Origin',
                    color_continuous_scale=px.colors.sequential.Inferno,
                    title='Average Total Cup Points by Country of Origin')

fig.show()
```

==========

13) Overall Insights Obtained from Analysis :

- *Summary of key insights and findings obtained from the analysis.*

Key Factors :

🚀 **Key Factors Affecting the Quatity of Coffee :**

💥 By doing EDA, I conclude that Quality measures 'Flavor', 'Aftertaste', 'Balance', 'Body', 'Moisture' and 'Defects' affects the Quality of the Coffee.

- Quality Measures like Flavor, Aftertaste, Balance and Body have a positive effect
- Wherearea Moisture and Defects (Category.One.Defects & Category.One.Defects)

  🚀 **Coffee Bean Factors on Quality of Coffee :**

  💥 Among the factors that affect coffee quality, several stand out as particularly influential are Species, Processing Method, Country of Origin, In-Country Partner and Variety.

  🚀 **Understanding the Relationships between Variables :**

  💥 The Bean_Metadata and Farm_Metadata have a signification relationship between The Total Quality Score of the Coffee.
  -Processing.Method, In.Country.Partner and Country.of.Origin.
  💥 Specific farm attributes, such as altitude, have a impacts on coffee processing method (dry/roasting).
- Altitude have a inner relationship Moisture which gives impact.
- The Higher the Moisture is the method will change according to Washed/Wet and Dry/Roast.
  💥 Certain processing methods and regions consistently produce higher quality coffee.
- The regions using the Washed/ Wet processing method is the most and have a goodscore on the Coffee Quality.

Important Variables :

- Flavor, Aftertaste, Acidity, Body and Balance. Total.Cup.Points which I have used to find.
- Processing Methods, Variety, Country Of Origin and Altitude.

Dependencies between Columns :

- **Flavor and Aftertaste** are strongly correlated, including that improvements in the Aroma may lead to enhancements in Flavor.
- **Species and Country.of.Origin** have a great relationship between them.
- **Country.of.Origin and Processing.Method** are also strongly related which means different countries use different methods to process the coffee.
- which results directly on **Processing.Method and Flavor & Aftertaste** of the coffee.

Interesting Facts :

Processing.Method is the most important factor which impacts the Qualities of the Coffee except for the Quakers and Defects and Defect.

Recommendations for Improving Coffee Quality and Production Practices :

Based on the analysis conducted on the Coffee Quality Dataset, here are several recommendations for the clients aimed at improving coffee quality and production processes: **Recommendations for Improving Coffee Quality**

1. **Focus on Key Quality Attributes:**
   The analysis indicates that attributes such as Flavor, Aftertaste, Balance, and Body are strongly correlated with overall coffee quality. Emphasize improving these aspects through targeted cultivation and processing techniques.
2. **Optimize Processing Methods:**
   Different processing methods significantly impact coffee quality. Invest in training and infrastructure to adopt the processing methods that have shown to yield higher quality coffee. Experiment with various methods to find the optimal ones for specific coffee bean varieties.
3. **Leverage Regional Strengths:**
   Regions and farms with consistently high-quality coffee should be studied and emulated. Encourage knowledge sharing and best practices among farmers in different regions. Promote regional coffee as a premium product if it consistently meets high-quality standards.
4. **Reducing Defects:**
   I recommend implement the farming practices onreducing the Defects , Quakers and maintaing the moisture levels because it can lower the quality of the Coffee.
   1. **Market Segmentation:**
      Divided by the Overall Coffee Score on the Derived Metrix, I would suggest that segment the market based on quality and cater to different customer segments with appropriate pricing strategies. High-quality coffee can be marketed as a premium product, while other segments can focus on cost-effective production for mass markets.

Strategic Recommendations

1. **Research and Development:**
   Invest in R&D to explore new varieties of coffee beans that might offer better quality or resilience against diseases and climate change. Collaborative research with agricultural universities or institutions can yield valuable insights.
2. **Consumer Education:**
   Educate consumers about the different aspects of coffee quality and the efforts made to ensure high standards. This can help in building a loyal customer base willing to pay a premium for quality coffee.
3. **Technology Adoption:**
   Encourage the adoption of technology in various stages of coffee production. Precision agriculture, automation in processing, and advanced analytics can lead to more efficient and higher-quality production.

By focusing on these recommendations, clients can enhance the quality of their coffee, optimize production processes, and potentially increase their profitability. Continuous improvement and adaptation to new findings will be crucial in maintaining a competitive edge in the coffee industry.

==========

14) Conclusion

- *Final conclusions drawn from the analysis.*
- *Recommendations or next steps for further analysis or action.*

This project highlights the importance of thorough data analysis in understanding coffee quality. The findings provide actionable insights for coffee producers and stakeholders, enabling them to optimize their processes and improve the quality of their products.
For Further analysis, I would recommend explore additional factors and refine these insights, focus on defects and quakers etc.

——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ——— ⋆·☆·⋆ ———

CODES USED FOR THIS DOCUMENTATION ARE GIVEN BELOW

Importing Data and Required Packages

➽ Importing Pandas, Numpy, Matplotlib and Seaborns Library

In [1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
data = pd.read_csv("C:/Users/Admin/Downloads/coffeeQuality (1).csv")
```

In [5]:
```
data.head() #📣 inspecting the First 5 Rows of the dataframe
```

Out [5]:

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Catego
0	0	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0
1	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1
2	2	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0
3	3	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2
4	4	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2

5 rows × 44 columns

In [7]:
```
data.tail() #📣 inspecting the last 5 Rows of the dataframe
```

Out [7]:

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.
1334	1334	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	NaN	robustasa	NaN	...	Blue-Green	1

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.
1335	1335	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	NaN	robustasa	40	...	Blue-Green	0
1336	1336	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	global opportunity fund	795 meters	...	NaN	6
1337	1337	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN	...	Green	1
1338	1338	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politico	NaN	...	NaN	9

5 rows × 44 columns

In [9]:
```
data.sample(5)  #📣 inspecting the Random 5 Rows of the dataframe
```

Out [9]:

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Cat
1253	1253	Arabica	ipanema coffees	Brazil	fazenda capoeirnha	020/17	dry mill	002/1660/0080	ipanema coffees	890	...	Green	3
403	403	Arabica	j.andrade	Mexico	el pino	NaN	el pino, cosautlan de carvajal	0	café andrade, s.a. de c.v.	1100	...	Green	0
58	58	Arabica	juan luis alvarado romero	Guatemala	finca el limon	NaN	beneficio serben	11-853-155	unicafe	4650	...	Bluish-Green	1
1337	1337	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN	...	Green	1
612	612	Arabica	federacion nacional de cafeteros	Colombia	NaN	NaN	NaN	03/01/6894	federacion nacional de cafeteros	NaN	...	Green	9

5 rows × 44 columns

In [11]:
```
#📣 Checking the shape of the Data :
print("Number of Rows:",data.shape[0])
print("Number of Columns:",data.shape[1])
```

```
Number of Rows: 1339
Number of Columns: 44
```

In [13]:
```
#📣 to ensure the original data remains same during Data Cleaning
df = data.copy()
```

Data Checks to Perform before Data Cleaning (Data Exploration)

➽ Check for Duplicate Value

➽ Check for Check for Missing Values

➽ Check for DataType

➽ Check the number of Unique Value in every Columns

➽ Check Statistics of Dataset

In [16]:
```
#📣 Check for Duplicate Values :
duplicates = data.duplicated().sum()
print("Number of Duplicate Values:", duplicates)
```

```
Number of Duplicate Values: 0
```

In [18]:
```
#📣 Checking the Missing Values and DataTypes of Columns :
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1339 entries, 0 to 1338
Data columns (total 44 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              1339 non-null   int64
 1   Species                 1339 non-null   object
 2   Owner                   1332 non-null   object
 3   Country.of.Origin       1338 non-null   object
 4   Farm.Name               980 non-null    object
 5   Lot.Number              276 non-null    object
```

```
 6   Mill                    1021 non-null   object
 7   ICO.Number              1180 non-null   object
 8   Company                 1130 non-null   object
 9   Altitude                1113 non-null   object
 10  Region                  1280 non-null   object
 11  Producer                1107 non-null   object
 12  Number.of.Bags          1338 non-null   float64
 13  Bag.Weight              1339 non-null   object
 14  In.Country.Partner      1339 non-null   object
 15  Harvest.Year            1292 non-null   object
 16  Grading.Date            1339 non-null   object
 17  Owner.1                 1332 non-null   object
 18  Variety                 1113 non-null   object
 19  Processing.Method       1169 non-null   object
 20  Aroma                   1339 non-null   float64
 21  Flavor                  1339 non-null   float64
 22  Aftertaste              1339 non-null   float64
 23  Acidity                 1339 non-null   float64
 24  Body                    1339 non-null   float64
 25  Balance                 1339 non-null   float64
 26  Uniformity              1339 non-null   float64
 27  Clean.Cup               1339 non-null   float64
 28  Sweetness               1339 non-null   float64
 29  Cupper.Points           1339 non-null   float64
 30  Total.Cup.Points        1339 non-null   float64
 31  Moisture                1339 non-null   float64
 32  Category.One.Defects    1339 non-null   int64
 33  Quakers                 1338 non-null   float64
 34  Color                   1069 non-null   object
 35  Category.Two.Defects    1339 non-null   int64
 36  Expiration              1339 non-null   object
 37  Certification.Body      1339 non-null   object
 38  Certification.Address   1339 non-null   object
 39  Certification.Contact   1339 non-null   object
 40  unit_of_measurement     1339 non-null   object
 41  altitude_low_meters     1109 non-null   float64
 42  altitude_high_meters    1109 non-null   float64
 43  altitude_mean_meters    1109 non-null   float64
dtypes: float64(17), int64(3), object(24)
memory usage: 460.4+ KB
```

In [20]:
```
# 📣 Check the number of Unique Values in Every Column
print("Number of Unique Value in Every Columns :")
data.nunique()
```

```
Number of Unique Value in Every Columns :
```

Out [20]:
```
Unnamed: 0               1339
Species                     2
Owner                     315
Country.of.Origin          36
Farm.Name                 571
Lot.Number                227
Mill                      459
ICO.Number                845
Company                   281
Altitude                  396
Region                    356
Producer                  692
Number.of.Bags            134
Bag.Weight                 56
In.Country.Partner         27
Harvest.Year               46
Grading.Date              567
Owner.1                   319
Variety                    29
Processing.Method           5
Aroma                      37
Flavor                     35
Aftertaste                 35
Acidity                    31
Body                       33
Balance                    33
Uniformity                 10
Clean.Cup                  11
Sweetness                  17
Cupper.Points              42
Total.Cup.Points          180
Moisture                   23
Category.One.Defects       18
Quakers                    11
Color                       3
Category.Two.Defects       38
Expiration                566
Certification.Body         26
Certification.Address      32
Certification.Contact      29
unit_of_measurement         2
altitude_low_meters       198
altitude_high_meters      198
altitude_mean_meters      211
dtype: int64
```

In [22]:
```
data.describe(include='all')
```

Out [22]:

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Categ
count	1339.000000	1339	1332	1338	980	276	1021	1180	1130	1113	...	1069	1339
unique	NaN	2	315	36	571	227	459	845	281	396	...	3	NaN
top	NaN	Arabica	juan luis alvarado romero	Mexico	various	1	beneficio ixchel	0	unex guatemala, s.a.	1100	...	Green	NaN
freq	NaN	1311	155	236	47	18	90	79	86	43	...	870	NaN
mean	669.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	3.556
std	386.680316	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	5.312
min	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.000
25%	334.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.000

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Categ
50%	669.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	2.000
75%	1003.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	4.000
max	1338.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	55.00

11 rows × 44 columns

Insights gained from initial data exploration

- There are 1339 rows and 44 columns in the Dataset.
- From the info we conclude that Quality Measures Columns are all have Numerical Values, Bean Metadata are Categorical Values.
- Farm Metadata has both the Values - Categoriacal and Numerical Values.
- From the above information all mean of Quality Measures are close to each other - between 7.40 to 9.85 except Quakers(Defects) which is 0.17.
- All standard deviation of Quality Measures are close to each other - between 0.370064 to 0.832121 except Aroma which is 5.53.
- All Quality Measure Columns min-value starts from 0.00.

I) Data Cleaning

Handle missing values:

Identify Missing Values

In [29]:
```
data.isnull()
```

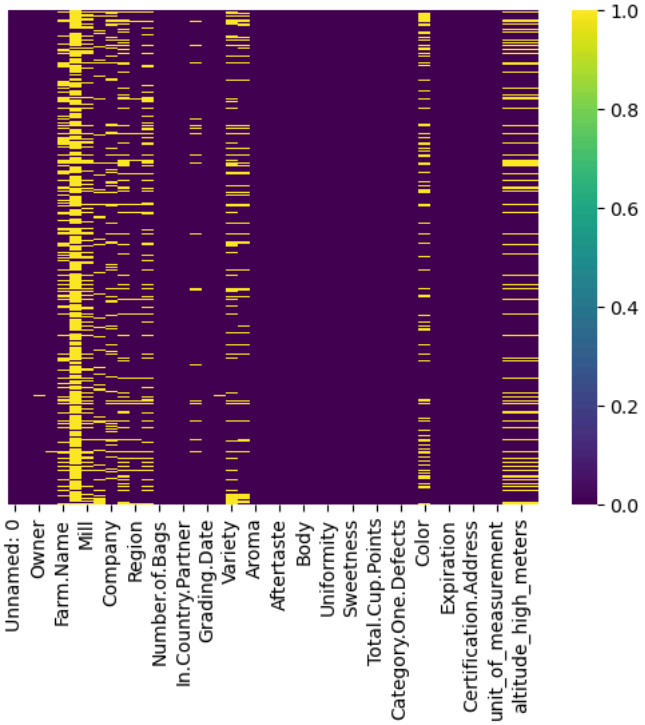
Out [29]:

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Def
0	False	False	False	False	False	True	False	False	False	False	...	False	False
1	False	False	False	False	False	True	False	False	False	False	...	False	False
2	False	False	False	False	False	True	True	True	True	False	...	True	False
3	False	False	False	False	False	True	False	True	False	False	...	False	False
4	False	False	False	False	False	True	False	False	False	False	...	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1334	False	False	False	False	False	False	False	True	False	True	...	False	False
1335	False	False	False	False	False	False	False	True	False	False	...	False	False
1336	False	False	False	False	False	True	False	True	False	False	...	True	False
1337	False	False	False	False	True	True	True	False	False	True	...	False	False
1338	False	False	False	False	True	True	True	True	False	True	...	True	False

1339 rows × 44 columns

In [31]:
```
sns.heatmap(data.isnull(),yticklabels=False,cmap = 'viridis') # 📣 Checking the null values in the heatmap
```

Out [31]: <Axes: >



In [42]:
```
# 📣 Check for missing values
missing_values = data.isna().sum()
```

```
display(Markdown(bold_text1))
print(missing_values)
```

```
<IPython.core.display.Markdown object>

Unnamed: 0                  0
Species                     0
Owner                       7
Country.of.Origin           1
Farm.Name                 359
Lot.Number               1063
Mill                      318
ICO.Number                159
Company                   209
Altitude                  226
Region                     59
Producer                  232
Number.of.Bags              1
Bag.Weight                  0
In.Country.Partner          0
Harvest.Year               47
Grading.Date                0
Owner.1                     7
Variety                   226
Processing.Method         170
Aroma                       0
Flavor                      0
Aftertaste                  0
Acidity                     0
Body                        0
Balance                     0
Uniformity                  0
Clean.Cup                   0
Sweetness                   0
Cupper.Points               0
Total.Cup.Points            0
Moisture                    0
Category.One.Defects        0
Quakers                     1
Color                     270
Category.Two.Defects        0
Expiration                  0
Certification.Body          0
Certification.Address       0
Certification.Contact       0
unit_of_measurement         0
altitude_low_meters       230
altitude_high_meters      230
altitude_mean_meters      230
dtype: int64
```

Comment:
We can observe from the above heatmap and in numbers as well that the missing value in the "Lot.Number" column. It is not necessary for EDA. So we can drop this column.
But Let's see the other columns as well. Let's drop the column which has the missing values is more then 30% of Total Values in the column.

In [81]:
```
null_percentage = (data.isnull().sum() / len(data)) * 100 # 📣 To Calculate the percentage of null values in e
formatted_null_percentage = null_percentage.apply(lambda x: "{:.2f}%".format(x))
display(Markdown(bold_text2)) # 📣 Printing the percentage of null values for each column
formatted_null_percentage
```

Out [81]:
```
<IPython.core.display.Markdown object>
Unnamed: 0                0.00%
Species                   0.00%
Owner                     0.52%
Country.of.Origin         0.07%
Farm.Name                26.81%
Lot.Number               79.39%
Mill                     23.75%
ICO.Number               11.87%
Company                  15.61%
Altitude                 16.88%
Region                    4.41%
Producer                 17.33%
Number.of.Bags            0.07%
Bag.Weight                0.00%
In.Country.Partner        0.00%
Harvest.Year              3.51%
Grading.Date              0.00%
Owner.1                   0.52%
Variety                  16.88%
Processing.Method        12.70%
Aroma                     0.00%
Flavor                    0.00%
Aftertaste                0.00%
Acidity                   0.00%
Body                      0.00%
Balance                   0.00%
Uniformity                0.00%
Clean.Cup                 0.00%
Sweetness                 0.00%
Cupper.Points             0.00%
Total.Cup.Points          0.00%
Moisture                  0.00%
Category.One.Defects      0.00%
Quakers                   0.07%
Color                    20.16%
Category.Two.Defects      0.00%
Expiration                0.00%
Certification.Body        0.00%
Certification.Address     0.00%
Certification.Contact     0.00%
unit_of_measurement       0.00%
altitude_low_meters      17.18%
altitude_high_meters     17.18%
altitude_mean_meters     17.18%
dtype: object
```

RESULTS

- The Columns which has more than 30% of null values will be removed = "Lot.Number".

- We also remove the columns because they contain null values and this column is not very useful for our analysis. = "Unnamed: 0", "ICO Number", "Certification.Address", "Certification.Contact", "unit_of_measurement", "altitude_low_meters", "altitude_high_meters" and "altitude_mean_meters".

In [85]:
```
a = df.shape[1]
```

└➤ Droping the Column

In [88]:
```
# Removing the Columns
df.drop('Lot.Number', axis=1, inplace=True) # null values are more than 30% (i.e 79.39%)
df.drop('Unnamed: 0', axis=1, inplace=True) # unncessary column
df.drop('ICO.Number', axis=1, inplace=True) # unncessary column
df.drop('Certification.Body', axis=1, inplace=True) # unncessary column
df.drop('Certification.Address', axis=1, inplace=True) # unncessary column
df.drop('Certification.Contact', axis=1, inplace=True) # unncessary column
df.drop('unit_of_measurement', axis=1, inplace=True) # unncessary column
df.drop('altitude_low_meters', axis=1, inplace=True) # unncessary column
df.drop('altitude_high_meters', axis=1, inplace=True) # unncessary column
df.drop('altitude_mean_meters', axis=1, inplace=True) # unncessary column
```

In [ ]:
```
df.columns # inspecting the DataFrame whether the columns have been dropped or not
```

In [90]:
```
print('No. of Columns after removing -',a)
print('No. of Columns after removing -',df.shape[1])
df.columns # inspecting the DataFrame whether the columns have been dropped or not
```

```
No. of Columns after removing - 44
No. of Columns after removing - 34
```

Out [90]:
```
Index(['Species', 'Owner', 'Country.of.Origin', 'Farm.Name', 'Mill', 'Company',
       'Altitude', 'Region', 'Producer', 'Number.of.Bags', 'Bag.Weight',
       'In.Country.Partner', 'Harvest.Year', 'Grading.Date', 'Owner.1',
       'Variety', 'Processing.Method', 'Aroma', 'Flavor', 'Aftertaste',
       'Acidity', 'Body', 'Balance', 'Uniformity', 'Clean.Cup', 'Sweetness',
       'Cupper.Points', 'Total.Cup.Points', 'Moisture', 'Category.One.Defects',
       'Quakers', 'Color', 'Category.Two.Defects', 'Expiration'],
      dtype='object')
```

In [95]:
```
# 📌Seperating Numerical Columns and Non-Numercal Columns for Missing Value Imputation/Replacing :
Numerical_Data = df.select_dtypes(include=['float64', 'int64']).columns
Non_Numerical_Data = df.select_dtypes(include=['object']).columns
```

In [132]:
```
# 📌 Printing all the Numeric & Non-Numeric Data and No. of Columns each data have :
display(Markdown(bold_text3))
print("🔷 No. of Numerical Columns =",len(Numerical_Data))
print(Numerical_Data)
display(Markdown(bold_text4))
print("🔷 No. of Non-Numerical Columns =",len(Non_Numerical_Data))
print(Non_Numerical_Data)
display((Markdown(bold_text5)),df.shape[1])
print('‾'*100)
```

```
<IPython.core.display.Markdown object>

🔷 No. of Numerical Columns = 16
Index(['Number.of.Bags', 'Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body',
       'Balance', 'Uniformity', 'Clean.Cup', 'Sweetness', 'Cupper.Points',
       'Total.Cup.Points', 'Moisture', 'Category.One.Defects', 'Quakers',
       'Category.Two.Defects'],
      dtype='object')

<IPython.core.display.Markdown object>

🔷 No. of Non-Numerical Columns = 18
Index(['Species', 'Owner', 'Country.of.Origin', 'Farm.Name', 'Mill', 'Company',
       'Altitude', 'Region', 'Producer', 'Bag.Weight', 'In.Country.Partner',
       'Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety',
       'Processing.Method', 'Color', 'Expiration'],
      dtype='object')

<IPython.core.display.Markdown object>
34
____________________________________________________________________________________________________
```

Comment : Before Replacing the Missing Values it is better to first Address the Inconsistencies in the necessary columns.

➽ Addressing Inconsistencies

└➤ Standardizing and Converting "Bag.Weight" Column into numerical column for missing value replacement

In [148]:
```
# Function to convert lbs to kg and handle uncertain values
def convert_to_kg(weight):
    if pd.isna(weight):
        return None
    parts = weight.split()
```

```
    if len(parts) == 2:
        value, unit = parts
        value = float(value)
        if unit == 'lbs':
            value *= 0.453592
    elif len(parts) == 1:
        value = float(parts[0])
        # Assuming ambiguous values are in kg
        # Modify this according to your specific context
    else:
        value = None  # handle unexpected formats
    return value

# Apply conversion
df['Bag.Weight'] = df['Bag.Weight'].apply(convert_to_kg)
```

In [144]:
```
display(Markdown(bold_text6))
bf1 = df['Bag.Weight']
bf1
```

Out [144]:
```
<IPython.core.display.Markdown object>
0       60 kg
1       60 kg
2           1
3       60 kg
4       60 kg
        ...
1334     2 kg
1335     2 kg
1336     1 kg
1337    5 lbs
1338    5 lbs
Name: Bag.Weight, Length: 1339, dtype: object
```

In [154]:
```
display(Markdown(bold_text7))
af1 = df['Bag.Weight']
af1
```

Out [154]:
```
<IPython.core.display.Markdown object>
0       60.00000
1       60.00000
2        1.00000
3       60.00000
4       60.00000
          ...
1334     2.00000
1335     2.00000
1336     1.00000
1337     2.26796
1338     2.26796
Name: Bag.Weight, Length: 1339, dtype: float64
```

↳➤ **Standardizing the "Harvest.Year" column for missing value replacement and further analysis.**

In [174]:
```
import re
def extract_year(value):
    # Ensure the value is treated as a string
    if pd.isna(value):
        return None
    value = str(value)
    # Regular expression to find year patterns
    year_pattern = re.compile(r'(\b\d{4}\b)')
    match = year_pattern.search(value)
    if match:
        return match.group(1)
    # Handle cases like 'Mar-10' (assume it's 2010)
    if re.match(r'^[A-Za-z]{3}-\d{2}$', value):
        return '20' + value[-2:]
    # Handle cases like 'May-August' (assuming no specific year provided, default to NaN or any specific year
    return None

# Apply the function to the 'Harvest.Year' column
df['Harvest.Year'] = df['Harvest.Year'].apply(extract_year)
```

In [172]:
```
display(Markdown(bold_text6))
bf2 = df['Harvest.Year'].iloc[2:19]
bf2
```

Out [172]:
```
<IPython.core.display.Markdown object>
2          NaN
3         2014
4         2014
5         2013
6         2012
7       Mar-10
8       Mar-10
9         2014
10        2014
11        2014
```

```
12                      2014
13    Sept 2009 - April 2010
14                    Mar-10
15                      2014
16                May-August
17                 2009/2010
18                      2015
Name: Harvest.Year, dtype: object
```

In [176]:
```
display(Markdown(bold_text7))
af2 = df['Harvest.Year'].iloc[2:19]
af2
```

```
<IPython.core.display.Markdown object>
2     None
3     2014
4     2014
5     2013
6     2012
7     2010
8     2010
9     2014
10    2014
11    2014
12    2014
13    2009
14    2010
15    2014
16    None
17    2009
18    2015
Name: Harvest.Year, dtype: object
```

Out [176]:

Standardizing the "Altitude" column for missing value replacement and further analysis.

In [198]:
```
# Since we already have - re library
# Function to clean and standardize the 'Altitude' column
def clean_and_standardize(altitude):
    if pd.isna(altitude):
        return np.nan  # Return NaN for missing values
    # Ensure the value is a string
    altitude = str(altitude)
    # Remove 'm', 'meters', and extra spaces
    altitude = re.sub(r'\s*m\s*|meters', '', altitude).strip()
    # Split by '-' and calculate the average if it's a range
    if '-' in altitude:
        try:
            parts = list(map(float, altitude.split('-')))
            return sum(parts) / len(parts)
        except ValueError:
            return np.nan
    # Convert single numeric values to float
    try:
        return float(altitude)
    except ValueError:
        return np.nan

# Apply the function to the 'Altitude' column
df['Altitude'] = df['Altitude'].apply(clean_and_standardize)
```

In [196]:
```
display(Markdown(bold_text6))
bf3 = df['Altitude']
bf3
```

```
<IPython.core.display.Markdown object>
0          1950-2200
1          1950-2200
2       1600 - 1800 m
3          1800-2200
4          1950-2200
            ...
1334             NaN
1335              40
1336       795 meters
1337             NaN
1338             NaN
Name: Altitude, Length: 1339, dtype: object
```

Out [196]:

In [200]:
```
display(Markdown(bold_text7))
af3 = df['Altitude']
af3
```

```
<IPython.core.display.Markdown object>
0       2075.0
1       2075.0
2       1700.0
3       2000.0
4       2075.0
         ...
1334       NaN
1335      40.0
1336       NaN
1337       NaN
```

Out [200]:

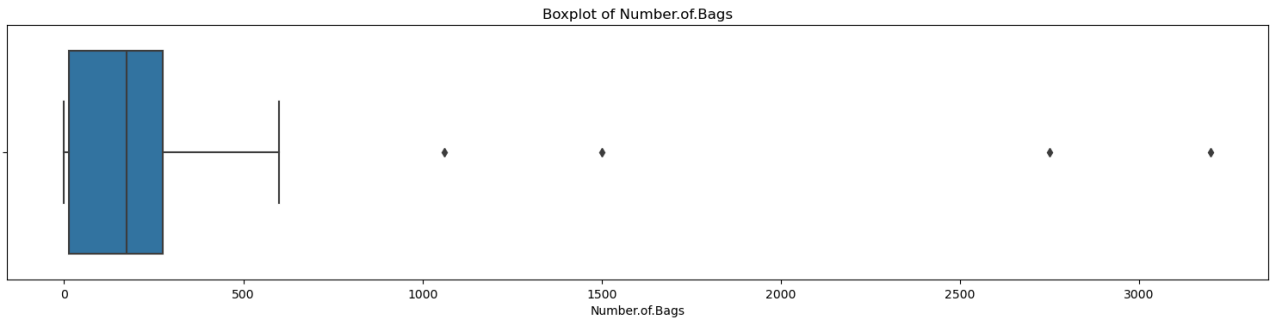
```
1338       NaN
Name: Altitude, Length: 1339, dtype: float64
```

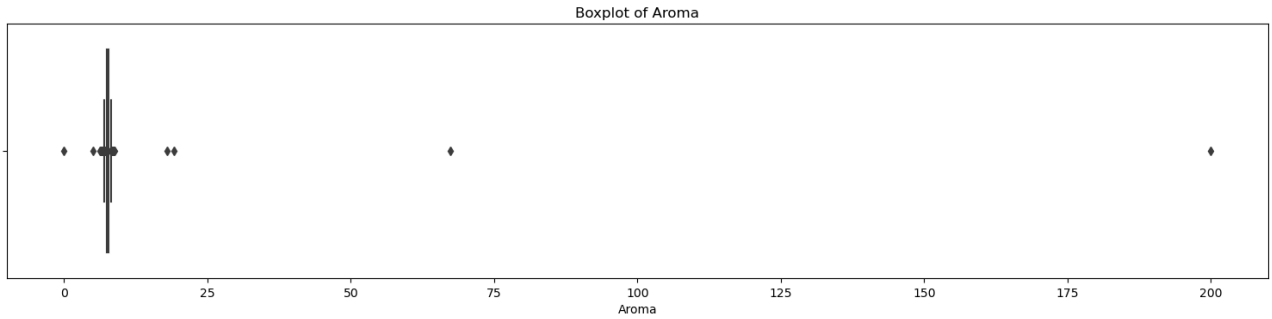
Comment : As we can see here that the Columns "Bag.Weight", "Harvest.Year" and "Altitude" is Standardized. Now we can Imputate/Replace Missing Values.

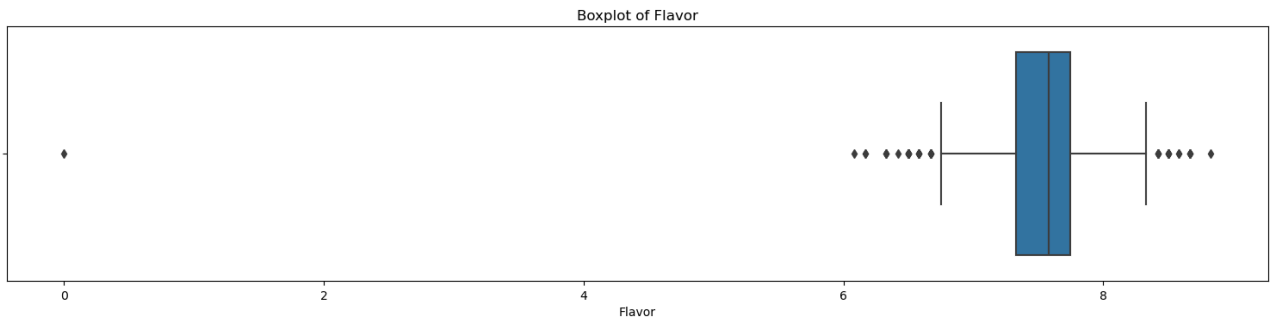
↳➤ Replacing/Imputating Missing Values

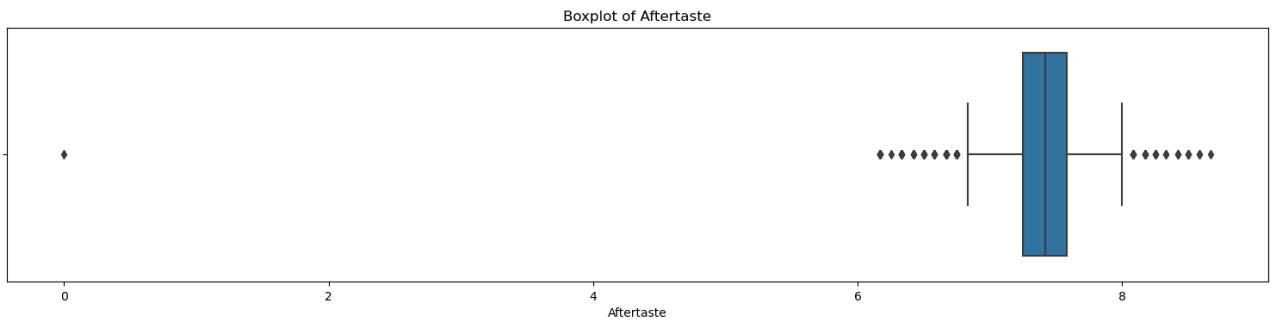
First let's Check if there are Outliers in the Data by using Boxplot :

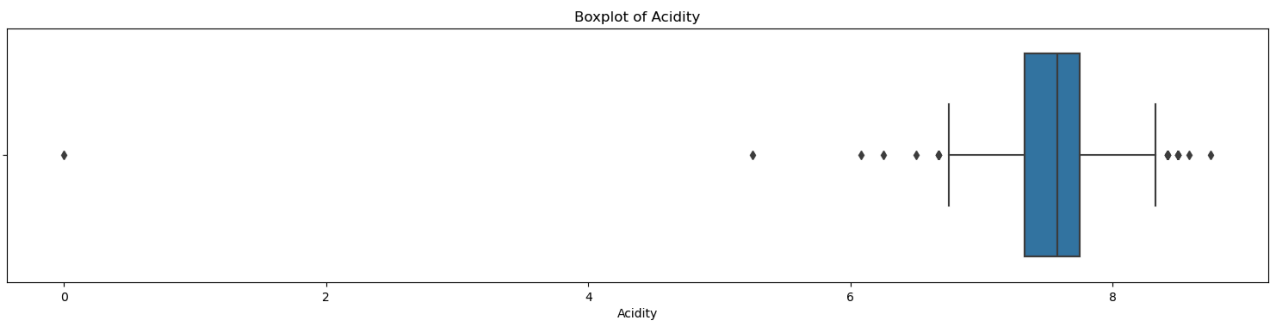
In [229]:
```
# For Numerical Data
# Plot boxplots for all numerical column :
plt.figure(figsize=(15, 60))  # Adjust the size to fit all boxplots
for i, col in enumerate(Numerical_Data, 1):
    plt.subplot(len(Numerical_Data), 1, i)
    sns.boxplot(data=df, x=col)
    plt.title(f'Boxplot of {col}')
    plt.xlabel(col)
    plt.tight_layout()
plt.show()
```


Boxplot of Number.of.Bags
0
500
1000
1500
2000
2500
3000
Number.of.Bags

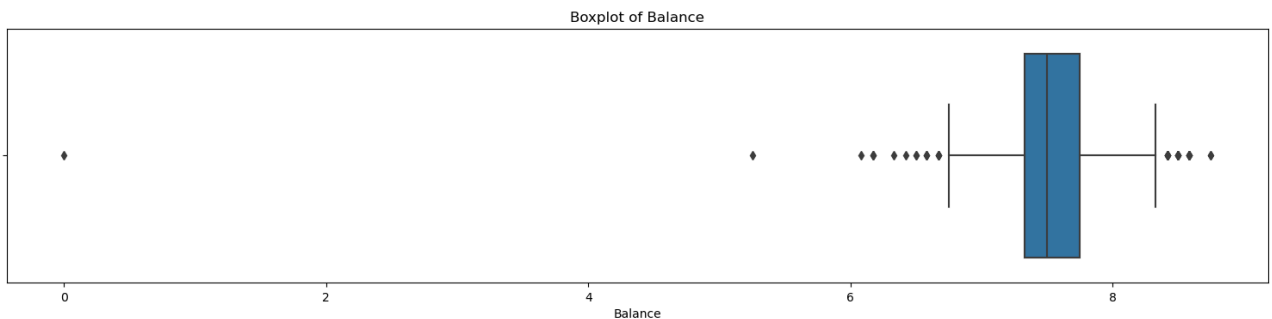

Boxplot of Aroma
0
25
50
75
100
125
150
175
200
Aroma

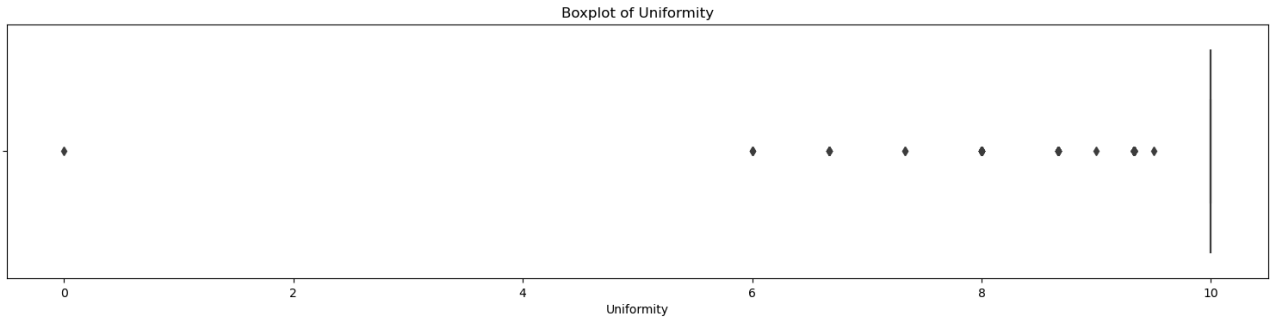

Boxplot of Flavor
0
2
4
6
8
Flavor

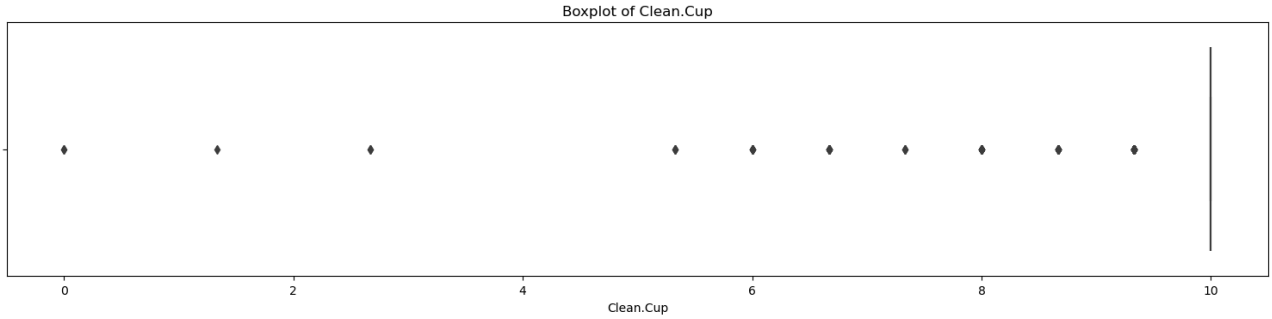

Boxplot of Aftertaste
0
2
4
6
8
Aftertaste

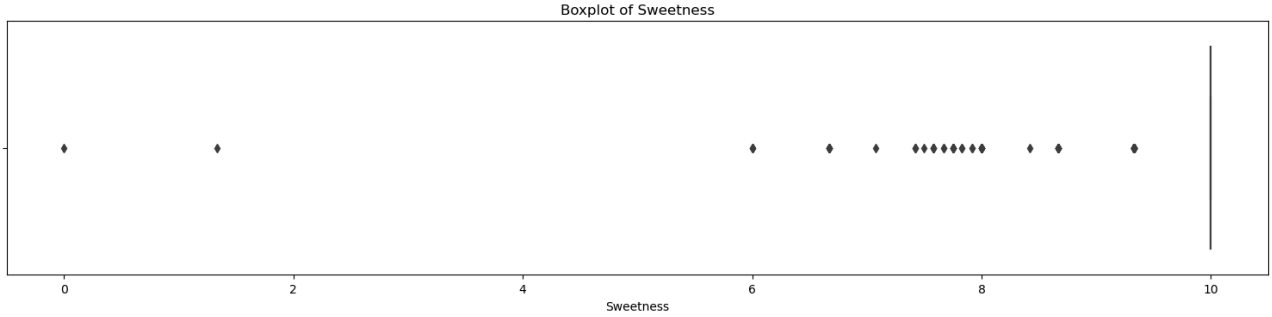

Boxplot of Acidity
0
2
4
6
8
Acidity

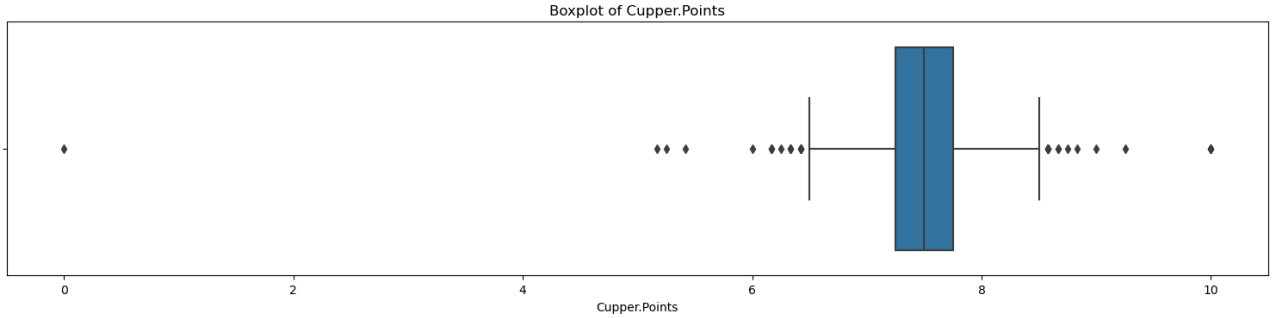

Boxplot of Body
0
2
4
6
8
Body

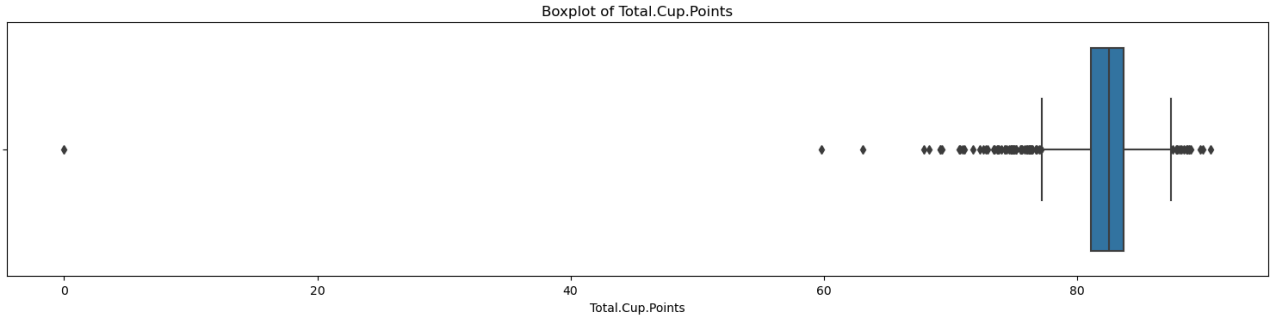

Boxplot of Balance
0
2
4
6
8
Balance

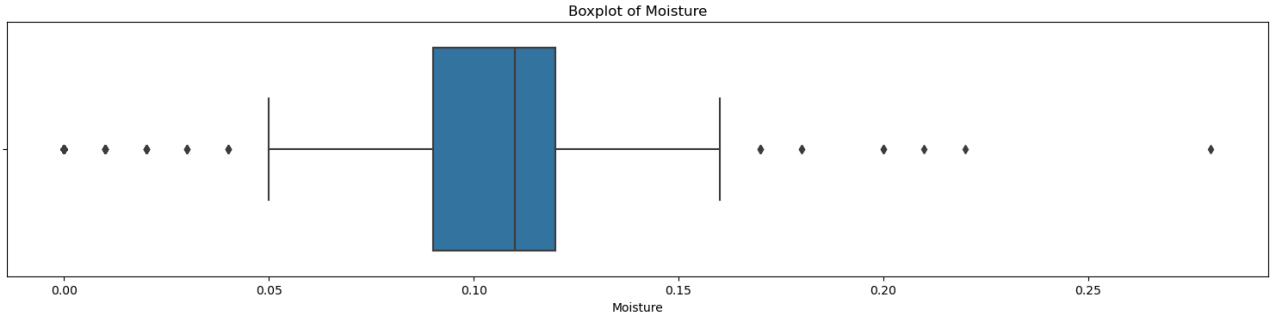

Boxplot of Uniformity
0
2
4
6
8
10
Uniformity

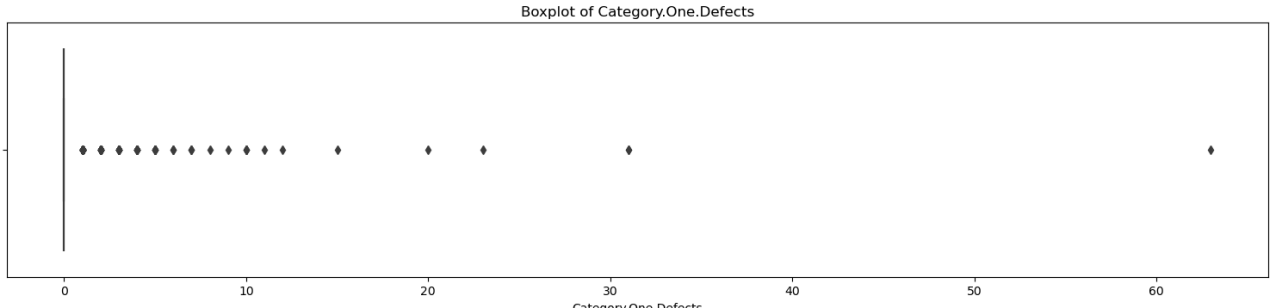

Boxplot of Clean.Cup
0
2
4
6
8
10
Clean.Cup

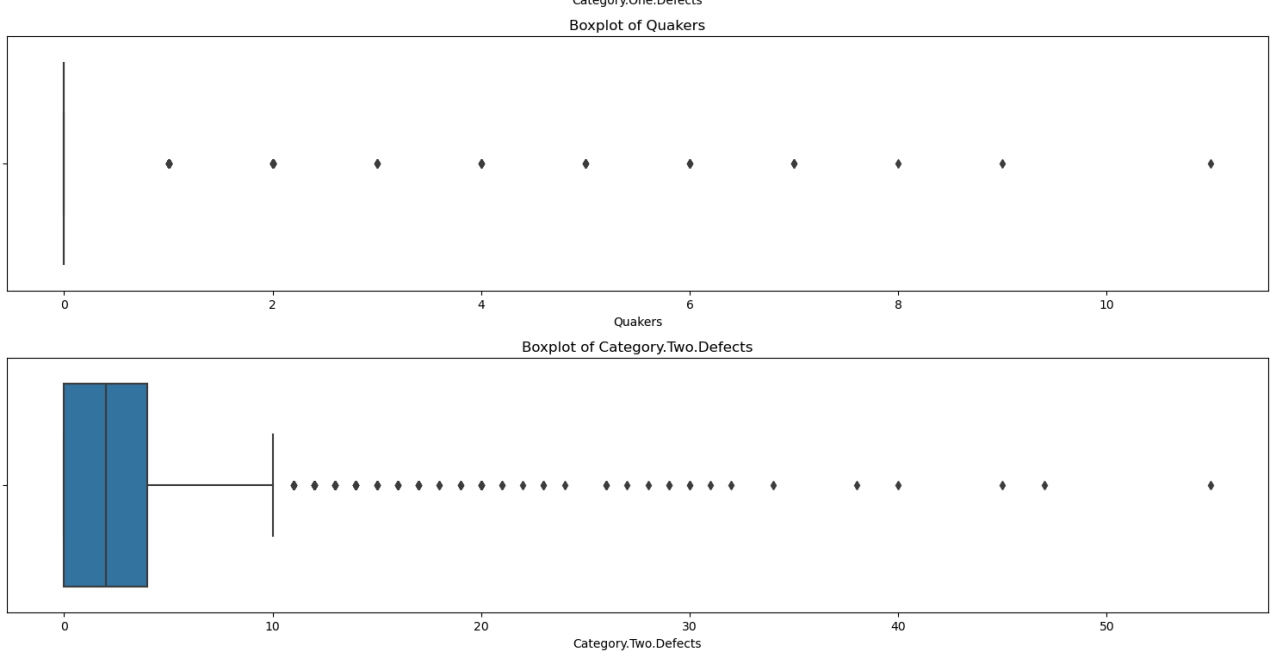

Boxplot of Sweetness
0
2
4
6
8
10
Sweetness


Boxplot of Cupper.Points
0
2
4
6
8
10
Cupper.Points


Boxplot of Total.Cup.Points
0
20
40
60
80
Total.Cup.Points


Boxplot of Moisture
0.00
0.05
0.10
0.15
0.20
0.25
Moisture


Boxplot of Category.One.Defects
0
10
20
30
40
50
60



Comment : All Columns has outliers. So it is replaced by median.

In [238]:
```
All_Columns = df.columns
All_Columns_count=len(df.columns)
print("All Columns:",All_Columns_count)
```

```
All Columns: 34
```

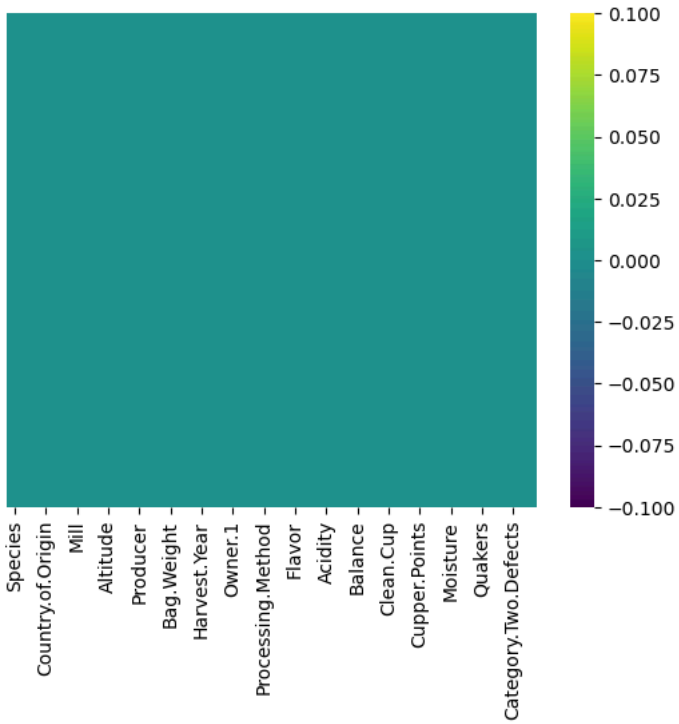
In [240]:
```
# Replacement by median for Numerical Data USING Median and Non-Numerical Data USING Mode :
for column in All_Columns:
    if df[column].dtype == 'object':
        mode_value = df[column].mode()[0]
        print(f"Filling missing values in column '{column}' with mode: {mode_value}")
        df[column].fillna(mode_value, inplace=True)
    else:
        median_value = df[column].median()
        print(f"Filling missing values in column '{column}' with median: {median_value}")
        df[column].fillna(median_value, inplace=True)
```

```
Filling missing values in column 'Species' with mode: Arabica
Filling missing values in column 'Owner' with mode: juan luis alvarado romero
Filling missing values in column 'Country.of.Origin' with mode: Mexico
Filling missing values in column 'Farm.Name' with mode: various
Filling missing values in column 'Mill' with mode: beneficio ixchel
Filling missing values in column 'Company' with mode: unex guatemala, s.a.
Filling missing values in column 'Altitude' with median: 1300.0
Filling missing values in column 'Region' with mode: huila
Filling missing values in column 'Producer' with mode: La Plata
Filling missing values in column 'Number.of.Bags' with median: 175.0
Filling missing values in column 'Bag.Weight' with median: 45.3592
Filling missing values in column 'In.Country.Partner' with mode: Specialty Coffee Association
Filling missing values in column 'Harvest.Year' with mode: 2012
Filling missing values in column 'Grading.Date' with mode: July 11th, 2012
Filling missing values in column 'Owner.1' with mode: Juan Luis Alvarado Romero
Filling missing values in column 'Variety' with mode: Caturra
Filling missing values in column 'Processing.Method' with mode: Washed / Wet
Filling missing values in column 'Aroma' with median: 7.58
Filling missing values in column 'Flavor' with median: 7.58
Filling missing values in column 'Aftertaste' with median: 7.42
Filling missing values in column 'Acidity' with median: 7.58
Filling missing values in column 'Body' with median: 7.5
Filling missing values in column 'Balance' with median: 7.5
Filling missing values in column 'Uniformity' with median: 10.0
Filling missing values in column 'Clean.Cup' with median: 10.0
Filling missing values in column 'Sweetness' with median: 10.0
Filling missing values in column 'Cupper.Points' with median: 7.5
Filling missing values in column 'Total.Cup.Points' with median: 82.5
Filling missing values in column 'Moisture' with median: 0.11
Filling missing values in column 'Category.One.Defects' with median: 0.0
Filling missing values in column 'Quakers' with median: 0.0
Filling missing values in column 'Color' with mode: Green
Filling missing values in column 'Category.Two.Defects' with median: 2.0
Filling missing values in column 'Expiration' with mode: December 26th, 2014
```

Comment : The Missing Values are replaced with median and mode. Let's check in the Heatmap and numbers of missing values again.

In [281]:
```
sns.heatmap(df.isnull(),yticklabels=False,cmap = 'viridis') # checking the null values in the heatmap
```

Out [281]: <Axes: >



In [283]:
```
df.isna().sum() # checking if the null values have been replaced
```

Out [283]:
```
Species                0
Owner                  0
Country.of.Origin      0
Farm.Name              0
Mill                   0
Company                0
Altitude               0
Region                 0
Producer               0
Number.of.Bags         0
Bag.Weight             0
In.Country.Partner     0
Harvest.Year           0
Grading.Date           0
Owner.1                0
Variety                0
Processing.Method      0
Aroma                  0
Flavor                 0
Aftertaste             0
Acidity                0
Body                   0
Balance                0
Uniformity             0
Clean.Cup              0
Sweetness              0
Cupper.Points          0
Total.Cup.Points       0
Moisture               0
Category.One.Defects   0
Quakers                0
Color                  0
Category.Two.Defects   0
Expiration             0
dtype: int64
```

Result

- We can see here that there are no null values now in the data and it was replaced/imputated by Median and Mode of the columns according to their datatype.

➽ Outliers

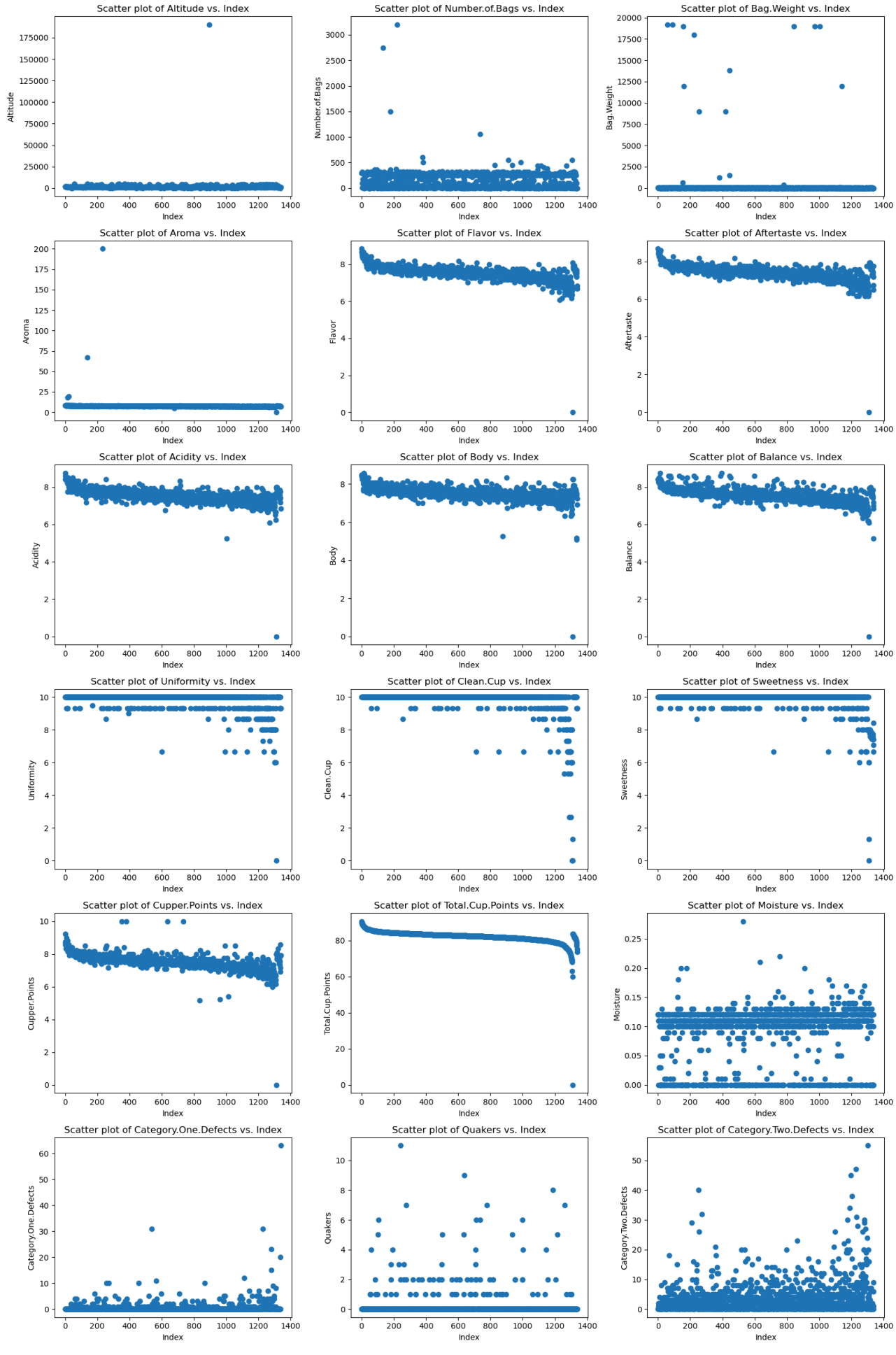
↳➤ Identifying and Removal of Outliers

In [288]:
```
df2 = df.copy()
```

In [292]:
```
display(Markdown(bold_text8))
# Identify numerical columns
numerical_columns = df.select_dtypes(include='number').columns
# Determine the number of rows and columns for the grid
n_cols = 3
n_rows = int(np.ceil(len(numerical_columns) / n_cols))
# Create subplots
fig, axes = plt.subplots(n_rows, n_cols, figsize=(16, n_rows * 4))
axes = axes.flatten()
# Plot scatter plots in the grid
for idx, col in enumerate(numerical_columns):
    ax = axes[idx]
    ax.scatter(df.index, df[col])
```

```
    ax.set_title(f'Scatter plot of {col} vs. Index')
    ax.set_xlabel('Index')
    ax.set_ylabel(col)
# Remove any empty subplots
for i in range(len(numerical_columns), len(axes)):
    fig.delaxes(axes[i])

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

<IPython.core.display.Markdown object>



Comment :

- From the above ScatterPlot we can able to see the outliers present in the data.
- The Category.Two.Defects havethe most number of outliers in the Dataset

Z-Score Test

Using zscore to find the Outliers in the important columns for Anaysis

In [540]:
```
from scipy import stats
# Calculate z-scores for the cleaned DataFrame
zscore1 = np.abs(stats.zscore(df2['Number.of.Bags']))
zscore2 = np.abs(stats.zscore(df2['Bag.Weight']))
zscore3 = np.abs(stats.zscore(df2['Altitude']))
zscore4 = np.abs(stats.zscore(df2['Aroma']))
zscore5 = np.abs(stats.zscore(df2['Flavor']))
zscore6 = np.abs(stats.zscore(df2['Aftertaste']))
zscore7 = np.abs(stats.zscore(df2['Acidity']))
zscore8 = np.abs(stats.zscore(df2['Clean.Cup']))
zscore9 = np.abs(stats.zscore(df2['Category.One.Defects']))
zscore10 = np.abs(stats.zscore(df2['Category.Two.Defects']))
```

In [542]:
```
threshold = 3
A=np.where(zscore1>3)
B=np.where(zscore2>3)
C=np.where(zscore3>3)
D=np.where(zscore4>3)
E=np.where(zscore5>3)
F=np.where(zscore6>3)
G=np.where(zscore7>3)
H=np.where(zscore8>3)
I=np.where(zscore9>3)
J=np.where(zscore10>3)
```

In [556]:
```
display(Markdown(bold_text9))
print('✔️ Number.of.Bags :',A)
print('✔️ Bag.Weight :',B)
print('✔️ Altitude :',C)
print('✔️ Aroma :',D)
print('✔️ Flavor :',E)
print('✔️ Aftertaste :',F)
print('✔️ Acidity :',G)
print('✔️ Clean.Cup :',H)
print('✔️ Category.One.Defects :',I)
print('✔️ Category.Two.Defects :',J)
```

```
<IPython.core.display.Markdown object>

✔️ Number.of.Bags : (array([133, 179, 220, 734], dtype=int64),)
✔️ Bag.Weight : (array([  59,   88,  158,  159,  224,  255,  420,  443,  843,  974, 1005,
       1139], dtype=int64),)
✔️ Altitude : (array([896], dtype=int64),)
✔️ Aroma : (array([139, 234], dtype=int64),)
✔️ Flavor : (array([   0, 1230, 1245, 1304, 1310], dtype=int64),)
✔️ Aftertaste : (array([   0, 1230, 1245, 1283, 1284, 1289, 1303, 1304, 1305, 1310],
      dtype=int64),)
✔️ Acidity : (array([   0, 1002, 1269, 1308, 1310], dtype=int64),)
✔️ Clean.Cup : (array([ 712,  851, 1006, 1171, 1219, 1259, 1270, 1277, 1278, 1279, 1280,
       1282, 1285, 1288, 1289, 1290, 1291, 1295, 1296, 1297, 1298, 1299,
       1300, 1301, 1302, 1306, 1307, 1308, 1309, 1310], dtype=int64),)
✔️ Category.One.Defects : (array([ 256,  271,  457,  536,  564,  866, 1113, 1229, 1279, 1280, 1291,
       1337, 1338], dtype=int64),)
✔️ Category.Two.Defects : (array([ 209,  250,  256,  271,  357,  518,  537,  797,  864, 1091, 1097,
       1154, 1172, 1176, 1179, 1188, 1190, 1195, 1203, 1229, 1232, 1239,
       1269, 1276, 1277, 1279, 1282, 1289, 1299, 1302, 1306], dtype=int64),)
```

↳➤ Removal of Outliers

In [590]:
```
# List of indices to drop
indices_of_outliers = [133, 179, 220, 734, 59, 88, 158, 224, 255, 420, 443, 843, 974, 1005, 1139, 896, 139, 234
# Drop the rows with the specified indices
df2 = df2.drop(index=indices_of_outliers)
```

Out [590]:

	Species	Owner	Country.of.Origin	Farm.Name	Mill	Company	Altitude	Region	Producer	Number.of.Bags	...	Sweetness
0	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	10.0
1	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	10.0

	Species	Owner	Country.of.Origin	Farm.Name	Mill	Company	Altitude	Region	Producer	Number.of.Bags	...	Sweetness
2	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	beneficio ixchel	unex guatemala, s.a.	1700.0	huila	La Plata	5.0	...	10.0
3	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	wolensu	yidnekachew debessa coffee plantation	2000.0	oromia	Yidnekachew Dabessa Coffee Plantation	320.0	...	10.0
4	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	10.0

5 rows × 35 columns

In [586]:
```
bfotl = df.shape
print("Shape of the Data Outlier Before Outlier Removal :",bfotl)
```

```
Shape of the Data Outlier Before Outlier Removal : (1339, 34)
```

In [594]:
```
afotl = df2.shape
print("Shape of the Data Outlier After Outlier Removal :",afotl)
```

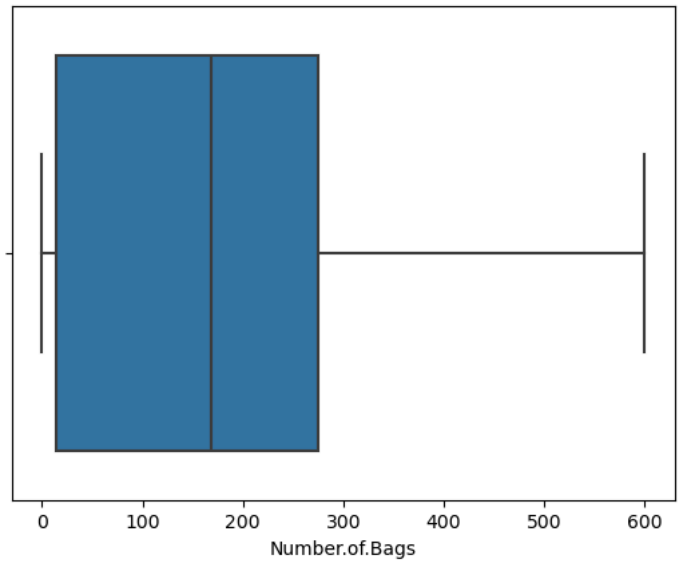
```
Shape of the Data Outlier After Outlier Removal : (1312, 35)
```

Results

- 27 Index Rows has been droped from the df by removal of Outliers from the data.
- For Example you can see the BoxPlot below that there no outliers after removing.

In [617]:
```
sns.boxplot(x=df2["Number.of.Bags"])
```

Out [617]:
```
<Axes: xlabel='Number.of.Bags'>
```



7) Obtaining Derived Metrics :

➢ *Description of derived metrics created to enhance the dataset.*

In [641]:
```
# Display the first few rows of the dataset
print("First few rows of the dataset:")
df2.head()
```

```
First few rows of the dataset:
```

Out [641]:

	Species	Owner	Country.of.Origin	Farm.Name	Mill	Company	Altitude	Region	Producer	Number.of.Bags	...	Sweetness
0	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	10.0
1	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	10.0
2	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	beneficio ixchel	unex guatemala, s.a.	1700.0	huila	La Plata	5.0	...	10.0

	Species	Owner	Country.of.Origin	Farm.Name	Mill	Company	Altitude	Region	Producer	Number.of.Bags	...	Sweetness
3	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	wolensu	yidnekachew debessa coffee plantation	2000.0	oromia	Yidnekachew Dabessa Coffee Plantation	320.0	...	10.0
4	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	10.0

5 rows × 35 columns

In [643]:
```
# Create a derived metric: Total Quality Score
quality_columns = ['Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance']
df2['Total_Quality_Score'] = df2[quality_columns].sum(axis=1)  # 📣 Sum of various quality measures
```

In [645]:
```
# Create a derived metric: Quality Ratio (e.g., Aroma to Flavor)
df2['Aroma_to_Flavor_Ratio'] = df2['Aroma'] / df2['Flavor']  # 📣 Ratio of Aroma to Flavor
```

In [653]:
```
# Create normalized quality measures
for column in quality_columns:
    df2[f'Normalized_{column}'] = (df2[column] - df2[column].mean()) / df2[column].std() # 📣 Normalize qualit
```

In [657]:
```
# Display the first few rows of the dataset with derived metrics
print("\nDataset with derived metrics:")
df2.head()
```

```
Dataset with derived metrics:
```

Out [657]:

	Species	Owner	Country.of.Origin	Farm.Name	Mill	Company	Altitude	Region	Producer	Number.of.Bags	...	Expiration
0	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	April 3rd, 2016
1	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	April 3rd, 2016
2	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	beneficio ixchel	unex guatemala, s.a.	1700.0	huila	La Plata	5.0	...	May 31st, 2011
3	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	wolensu	yidnekachew debessa coffee plantation	2000.0	oromia	Yidnekachew Dabessa Coffee Plantation	320.0	...	March 25th, 2016
4	Arabica	metad plc	Ethiopia	metad plc	metad plc	metad agricultural developmet plc	2075.0	guji-hambela	METAD PLC	300.0	...	April 3rd, 2016

5 rows × 43 columns

In [678]:
```
# Summary statistics of the new derived metrics
display(Markdown(bold_text11))
print(df2[['Total_Quality_Score', 'Aroma_to_Flavor_Ratio'] + [f'Normalized_{col}' for col in quality_columns]]
```

```
<IPython.core.display.Markdown object>

       Total_Quality_Score  Aroma_to_Flavor_Ratio  Normalized_Aroma  \
count          1312.000000            1312.000000      1.312000e+03
mean             45.113742               1.008812     -1.949660e-16
std               1.780924               0.057805      1.000000e+00
min              38.680000               0.655484     -4.704829e+00
25%              44.090000               0.989333     -3.182142e-01
50%              45.170000               1.000000     -1.827470e-02
75%              46.080000               1.022667      3.004110e-01
max              59.790000               2.350061      2.176483e+01

       Normalized_Flavor  Normalized_Aftertaste  Normalized_Acidity  \
count       1.312000e+03           1.312000e+03        1.312000e+03
mean       -9.531671e-16           1.213122e-15        1.819683e-15
std         1.000000e+00           1.000000e+00        1.000000e+00
min        -4.248156e+00          -3.543280e+00       -7.184642e+00
25%        -5.783715e-01          -4.514389e-01       -6.635418e-01
50%         1.555854e-01           3.523969e-02        1.202443e-01
75%         6.546761e-01           4.932901e-01        6.532188e-01
max         3.825370e+00           3.613759e+00        3.788363e+00

       Normalized_Body  Normalized_Balance
count     1.312000e+03        1.312000e+03
mean      1.126470e-15       -1.603054e-15
std       1.000000e+00        1.000000e+00
min      -7.926367e+00       -6.442797e+00
25%      -6.257690e-01       -5.517069e-01
50%      -7.416833e-02       -7.022352e-02
75%       4.774324e-01        6.378402e-01
max       3.430119e+00        3.470095e+00
```

II) Exploratory Data Analysis (EDA)

TASKS TO BE PERFORMED :

• Univariate Analysis:

Explore distributions and summary statistics of individual variables such as quality measures (aroma, flavor, etc.), bean metadata, and farm metadata.

• Bivariate Analysis:

Investigate relationships between pairs of variables, examining correlations or associations between quality measures, bean characteristics, and farm attributes.

• Multivariate Analysis:

Explore interactions and dependencies among multiple variables using techniques like clustering or dimensionality reduction.

➽ Univariate Analysis

└➤ EDA - Univariate Analysis :

- *Insights gained from univariate analysis, including visualizations of individual variables.*
- *Explore distributions and summary statistics of individual variables such as quality measures (aroma, flavor, etc.), bean metadata, and farm metadata.*

>> Summary Statistics:

In [731]:
```
quality_measures.describe()
```

Out [731]:

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Cupper.Points	Clean.Cup	Sweetnes
count	1312.000000	1312.000000	1312.000000	1312.000000	1312.000000	1312.000000	1312.000000	1312.000000	1312.000000	1312.00000
mean	7.589748	7.527005	7.407691	7.541646	7.522858	7.524794	9.846220	7.508194	9.855602	9.870511
std	0.533441	0.340619	0.349306	0.318965	0.308194	0.353076	0.472036	0.426657	0.644243	0.499521
min	5.080000	6.080000	6.170000	5.250000	5.080000	5.250000	6.000000	5.170000	0.000000	6.000000
25%	7.420000	7.330000	7.250000	7.330000	7.330000	7.330000	10.000000	7.250000	10.000000	10.000000
50%	7.580000	7.580000	7.420000	7.580000	7.500000	7.500000	10.000000	7.500000	10.000000	10.000000
75%	7.750000	7.750000	7.580000	7.750000	7.670000	7.750000	10.000000	7.750000	10.000000	10.000000
max	19.200000	8.830000	8.670000	8.750000	8.580000	8.750000	10.000000	10.000000	10.000000	10.000000

In [734]:
```
bean_metadata.describe()
```

Out [734]:

	Species	Processing.Method	Color
count	1312	1312	1312
unique	2	5	3
top	Arabica	Washed / Wet	Green
freq	1285	967	1116

In [748]:
```
farm_metadata.describe(include='all')
```

Out [748]:

	Owner	Country.of.Origin	Farm.Name	Mill	Company	Altitude	Region
count	1312	1312	1312	1312	1312	1312.000000	1312
unique	309	36	563	451	276	NaN	349
top	juan luis alvarado romero	Mexico	various	beneficio ixchel	unex guatemala, s.a.	NaN	huila
freq	158	234	399	401	293	NaN	169
mean	NaN	NaN	NaN	NaN	NaN	1413.152965	NaN
std	NaN	NaN	NaN	NaN	NaN	702.055083	NaN
min	NaN	NaN	NaN	NaN	NaN	0.000000	NaN
25%	NaN	NaN	NaN	NaN	NaN	1250.000000	NaN
50%	NaN	NaN	NaN	NaN	NaN	1300.000000	NaN
75%	NaN	NaN	NaN	NaN	NaN	1450.000000	NaN
max	NaN	NaN	NaN	NaN	NaN	5000.000000	NaN

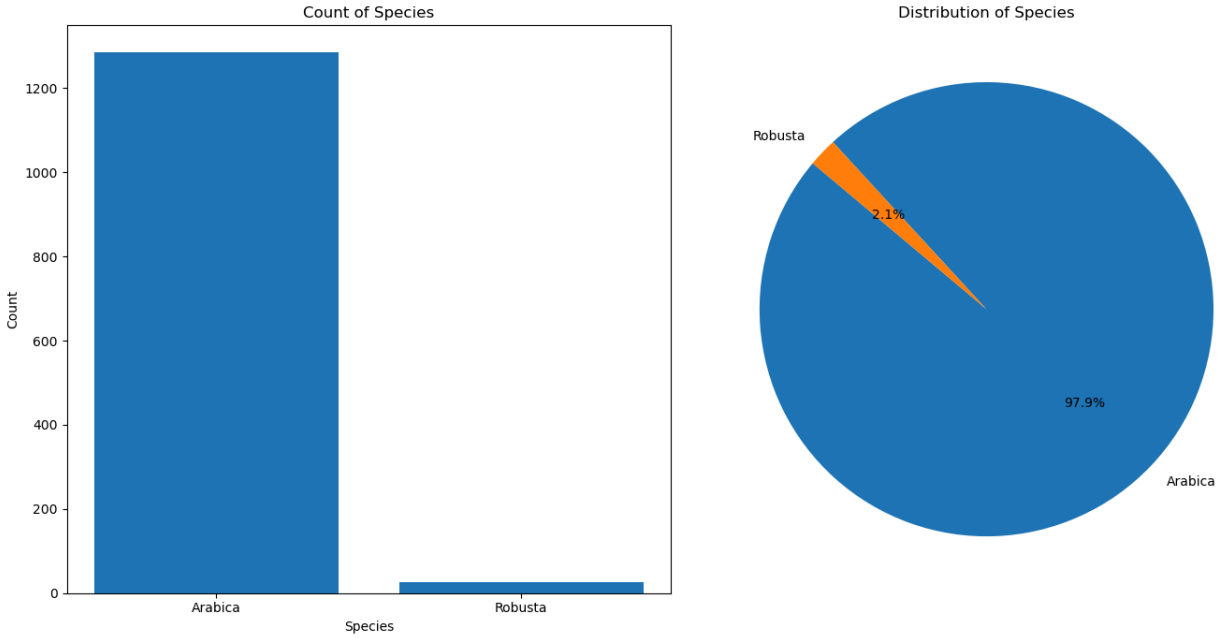
Comments : These are the Distribution of Quality Measures, Bean Metadata and Farm Metadata. The Visualizations of Univariate Analysis are given below.

In [750]:
```
# using count analysing the Univariate under Categorical value :
fre_tab=df2.groupby(['Species']).size().reset_index(name='Count').rename(columns={'Species':'Species'})
fre_tab1=df2.groupby(['Color']).size().reset_index(name='Count').rename(columns={'Color':'Color'})
fre_tab2=df2.groupby(['Processing.Method']).size().reset_index(name='Count').rename(columns={'Processing.Metho
fre_tab['Count%']= fre_tab['Count']/sum(fre_tab['Count'])*100
```

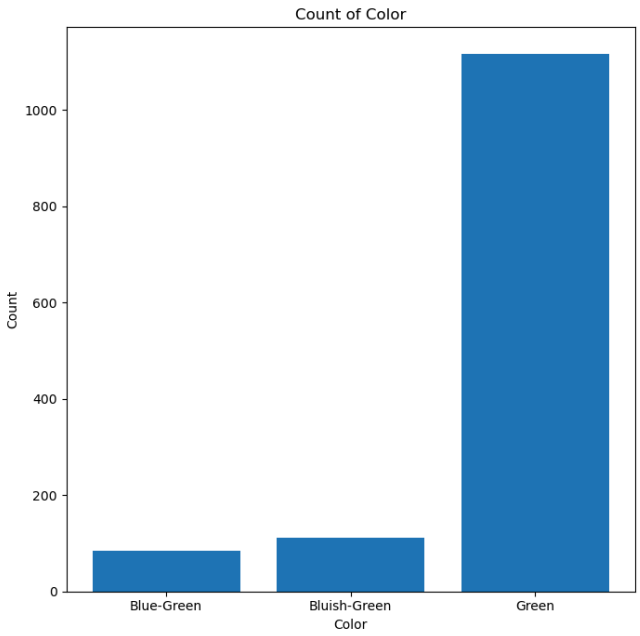
```
fre_tab1['Count%']= fre_tab1['Count']/sum(fre_tab1['Count'])*100
fre_tab2['Count%']= fre_tab2['Count']/sum(fre_tab2['Count'])*100
```

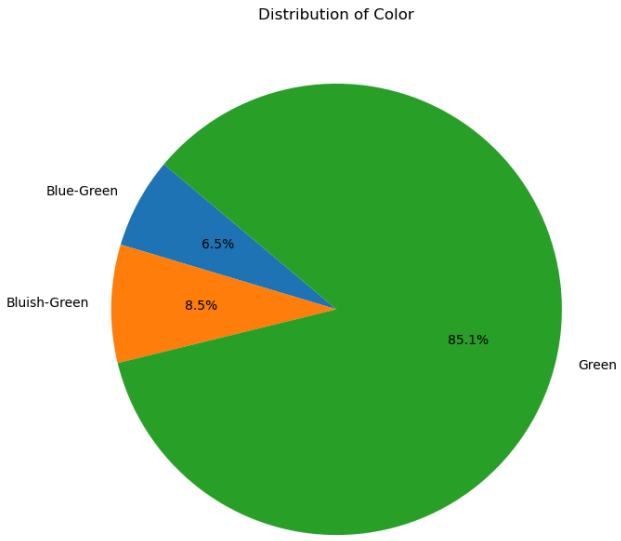
Visualisation of Distribution of Species, Color and Processing.Method of Coffee by Bar Chart and Pie Chart.

In [756]:
```
# Creating a figure with two subplots: one for the bar chart and one for the pie chart
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Bar chart
axs[0].bar(fre_tab['Species'], fre_tab['Count'])
axs[0].set_title('Count of Species')
axs[0].set_xlabel('Species')
axs[0].set_ylabel('Count')
# Pie chart
axs[1].pie(fre_tab['Count%'], labels=fre_tab['Species'], autopct='%1.1f%%', startangle=140)
axs[1].set_title('Distribution of Species')
# Display the charts
plt.tight_layout()
plt.show()
```

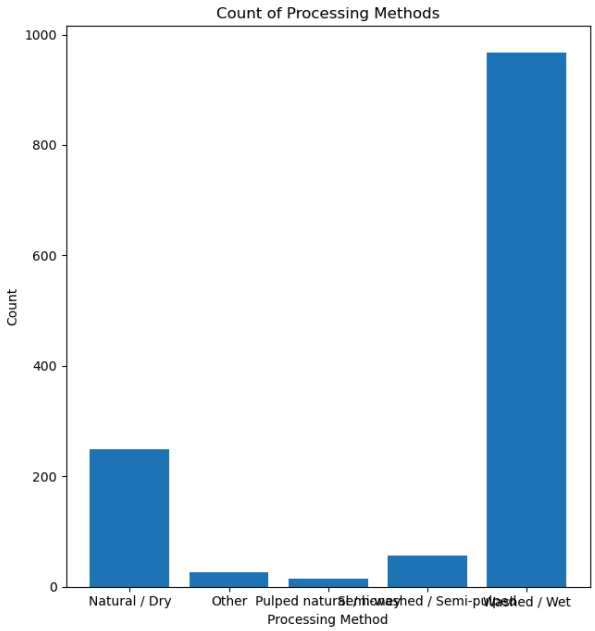


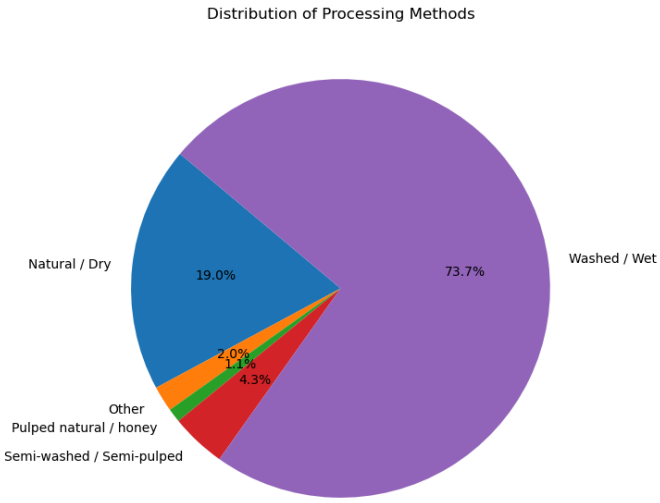
In [758]:
```
# Create a figure with two subplots: one for the bar chart and one for the pie chart
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Bar chart
axs[0].bar(fre_tab1['Color'], fre_tab1['Count'])
axs[0].set_title('Count of Color')
axs[0].set_xlabel('Color')
axs[0].set_ylabel('Count')
# Pie chart
axs[1].pie(fre_tab1['Count%'], labels=fre_tab1['Color'], autopct='%1.1f%%', startangle=140)
axs[1].set_title('Distribution of Color')
# Display the charts
plt.tight_layout()
plt.show()
```





In [760]:
```
# Create a figure with two subplots: one for the bar chart and one for the pie chart
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Bar chart
axs[0].bar(fre_tab2['Processing.Method'], fre_tab2['Count'])
axs[0].set_title('Count of Processing Methods')
axs[0].set_xlabel('Processing Method')
axs[0].set_ylabel('Count')
# Pie chart
axs[1].pie(fre_tab2['Count%'], labels=fre_tab2['Processing.Method'], autopct='%1.1f%%', startangle=140)
axs[1].set_title('Distribution of Processing Methods')
# Display the charts
plt.tight_layout()
plt.show()
```





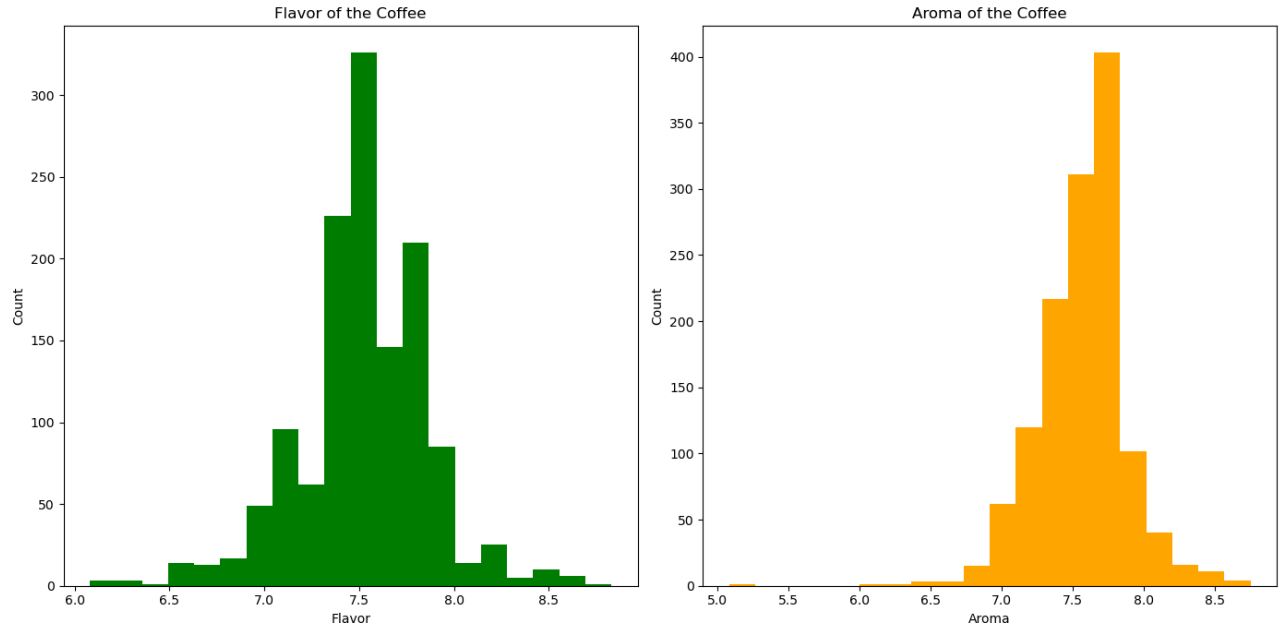
Visualisation of Count Distributions of Quality Measures of Coffee :

In [782]:
```
cleaned_df = df2[df2['Aroma'] <= 10]
```

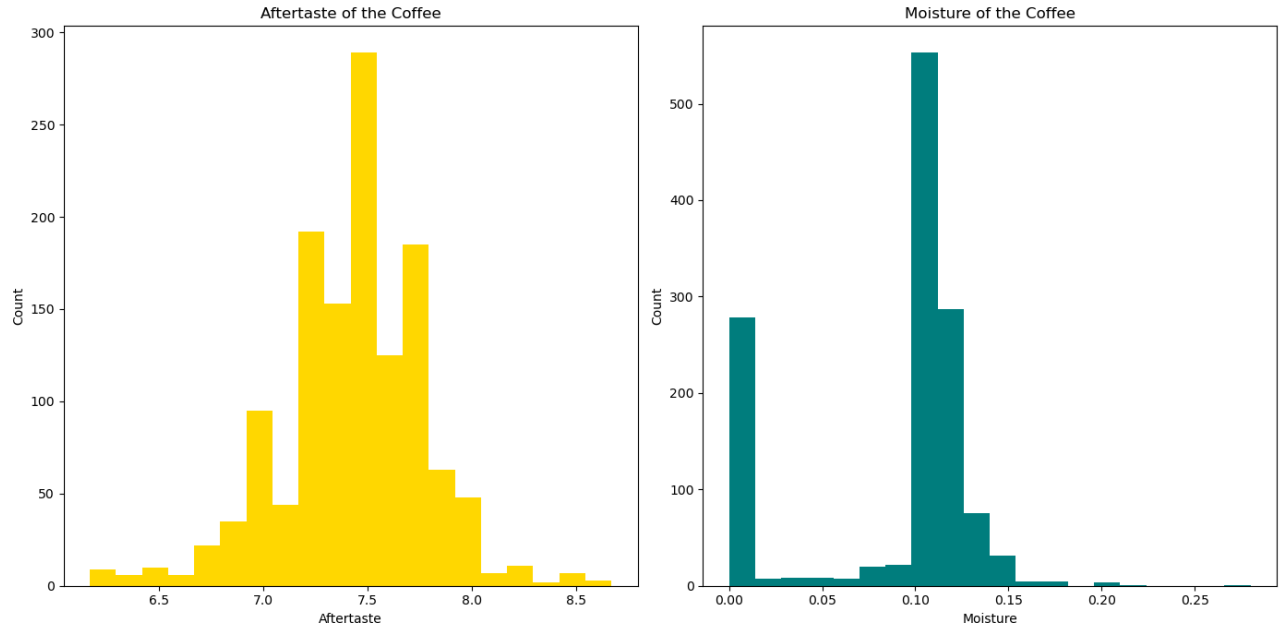
Histogram graphs of some Quality Measures :

In [784]:
```
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Plot the histogram for the 'Flavor' values
x = df2["Flavor"]
axs[0].hist(x, bins=20, color="green")
axs[0].set_title("Flavor of the Coffee")
axs[0].set_xlabel("Flavor")
axs[0].set_ylabel("Count")
# Plot the histogram for the 'Aroma' values
y = cleaned_df["Aroma"]
```

```
axs[1].hist(y, bins=20, color="orange")
axs[1].set_title("Aroma of the Coffee")
axs[1].set_xlabel("Aroma")
axs[1].set_ylabel("Count")
# Display the charts
plt.tight_layout()
plt.show()
```



In [786]:
```
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Plot the histogram for the 'Flavor' values
x = df2["Aftertaste"]
axs[0].hist(x, bins=20, color="gold")
axs[0].set_title("Aftertaste of the Coffee")
axs[0].set_xlabel("Aftertaste")
axs[0].set_ylabel("Count")
# Plot the histogram for the 'Aroma' values
y = df2["Moisture"]
axs[1].hist(y, bins=20, color="Teal")
axs[1].set_title("Moisture of the Coffee")
axs[1].set_xlabel("Moisture")
axs[1].set_ylabel("Count")
# Display the charts
plt.tight_layout()
plt.show()
```

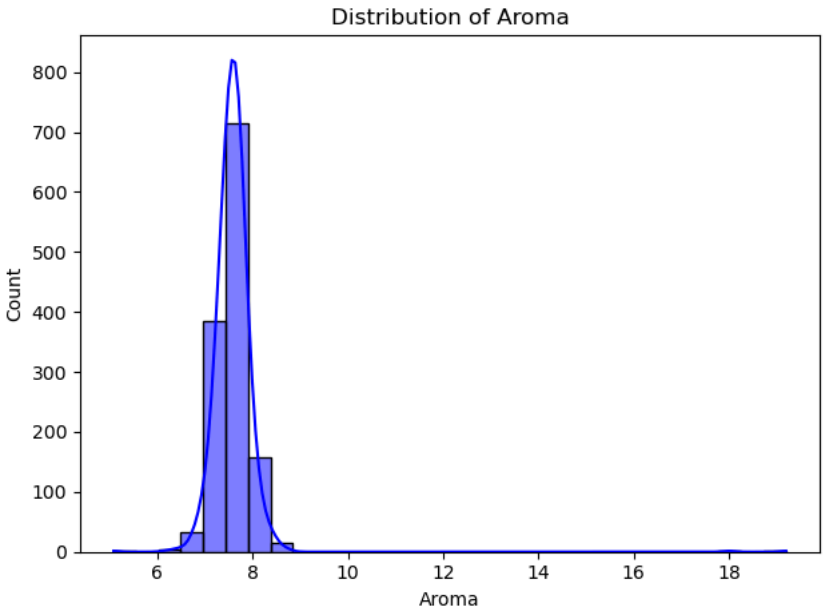


>> Distributions :

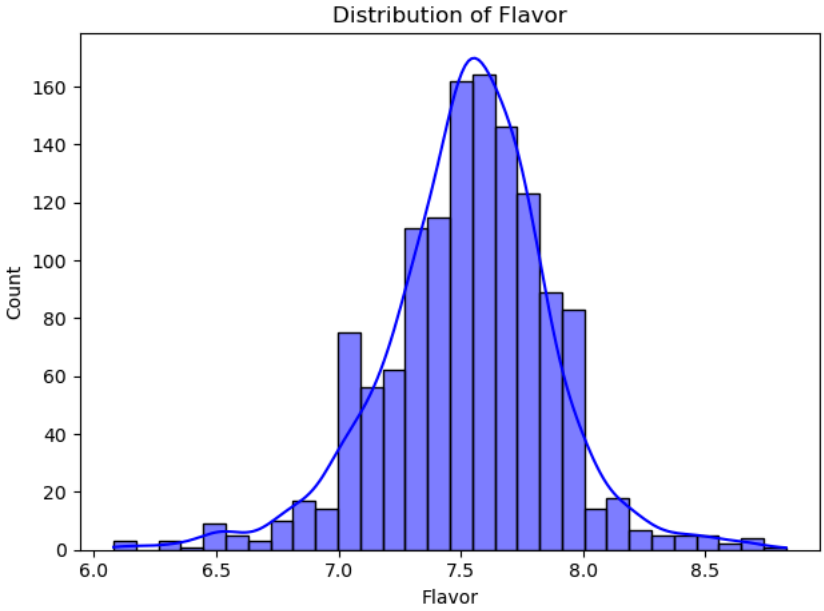
In [791]:
```
# Plot histograms for quality measures
for measure in quality_measures:
    plt.figure()
```

```
    sns.histplot(df2[measure], kde=True, color='blue', bins=30)
    plt.title(f'Distribution of {measure}')
    plt.tight_layout()
    plt.show()
```

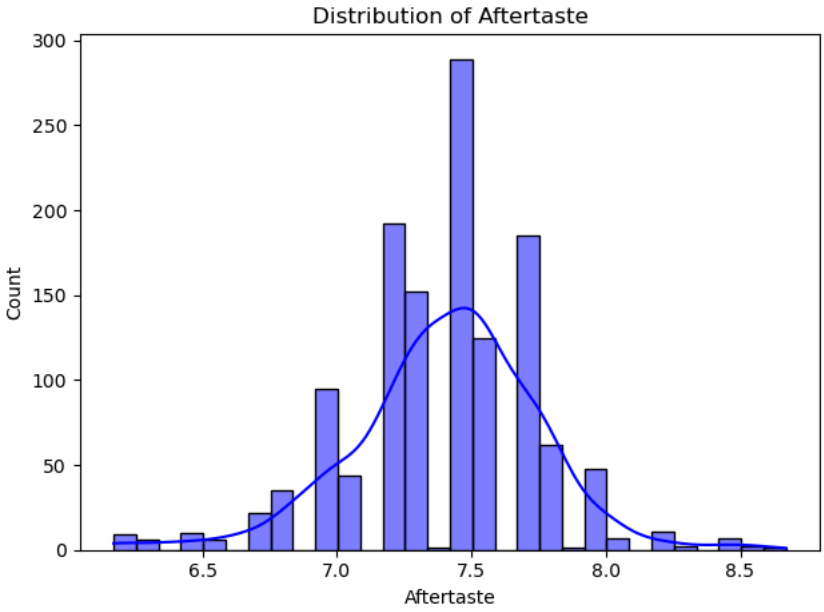
```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



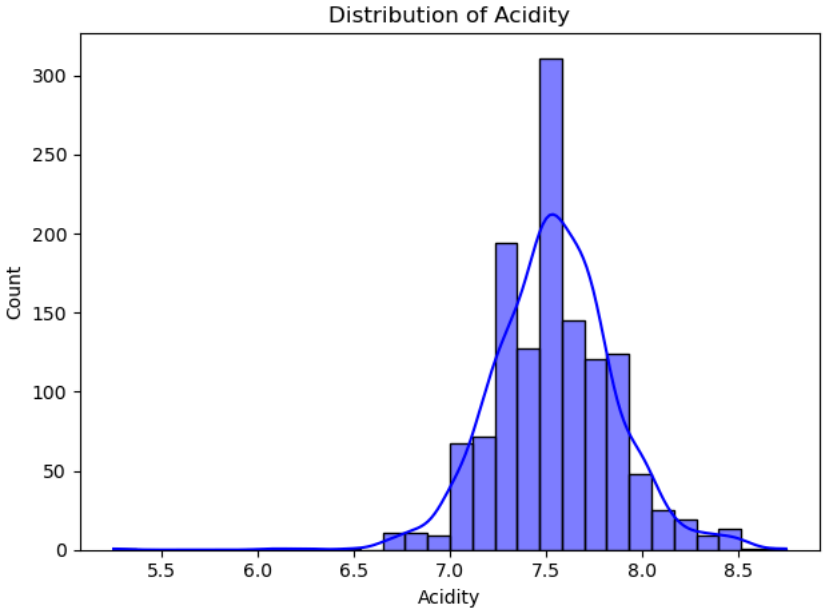

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



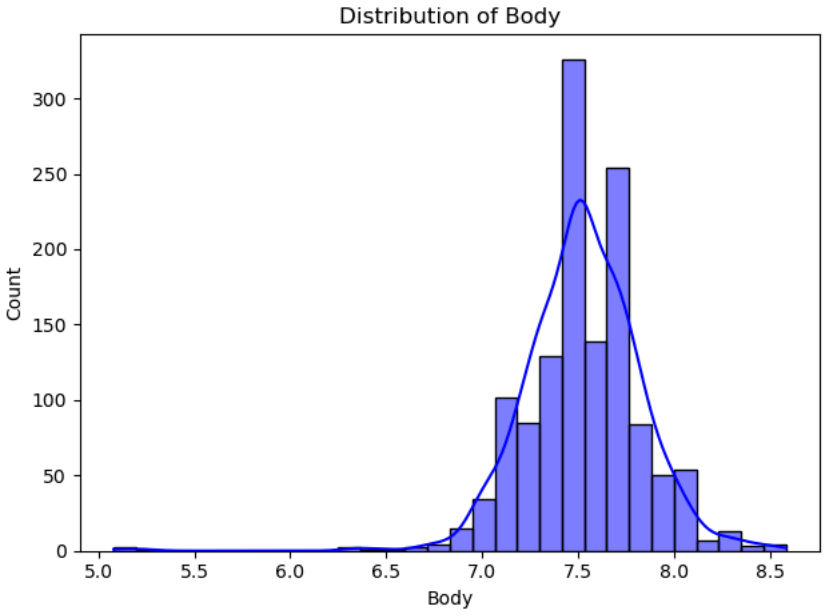

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



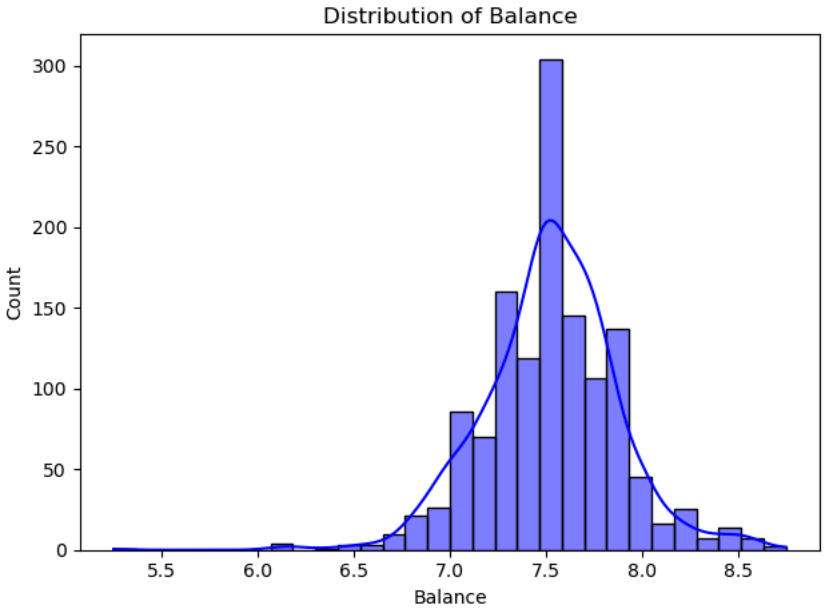

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



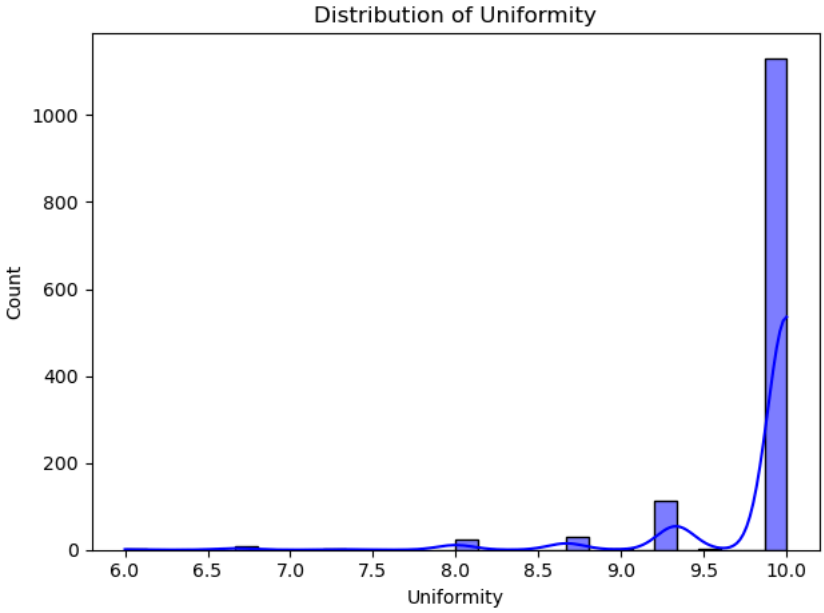

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



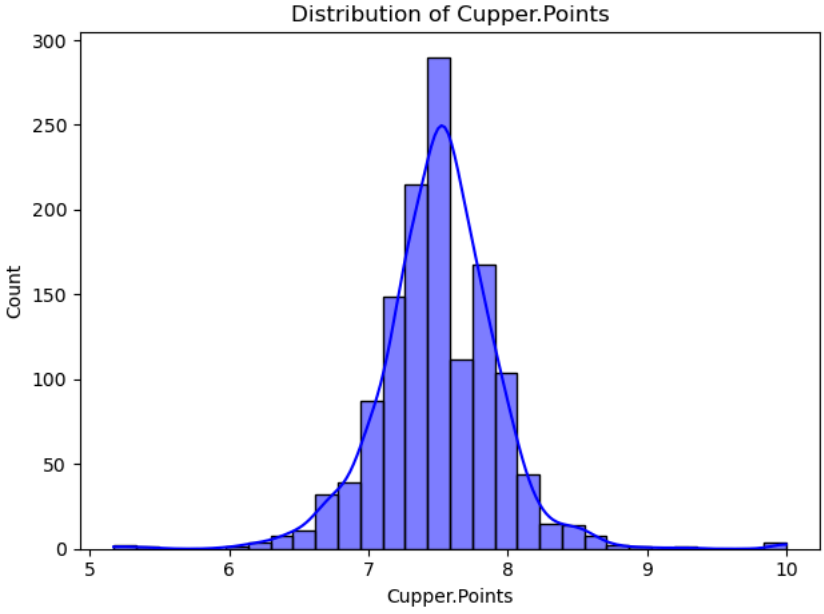

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



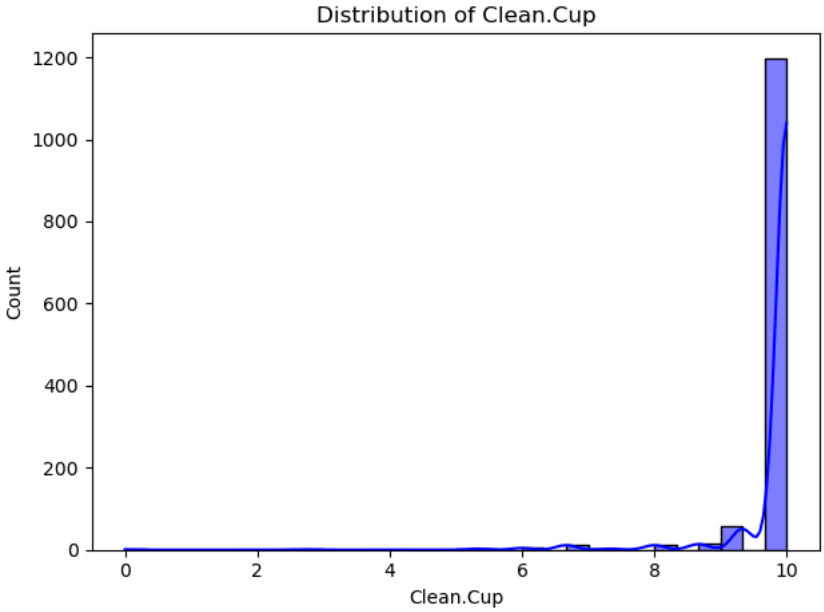

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



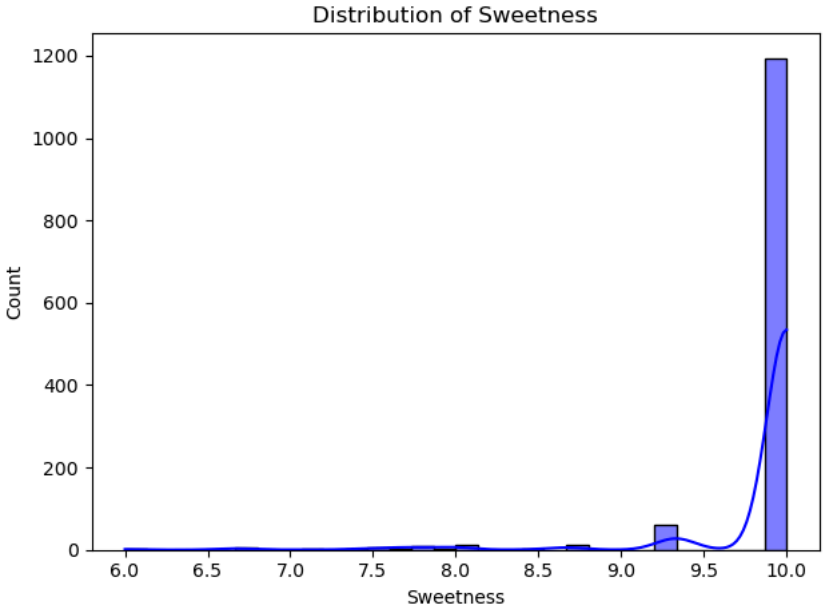

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



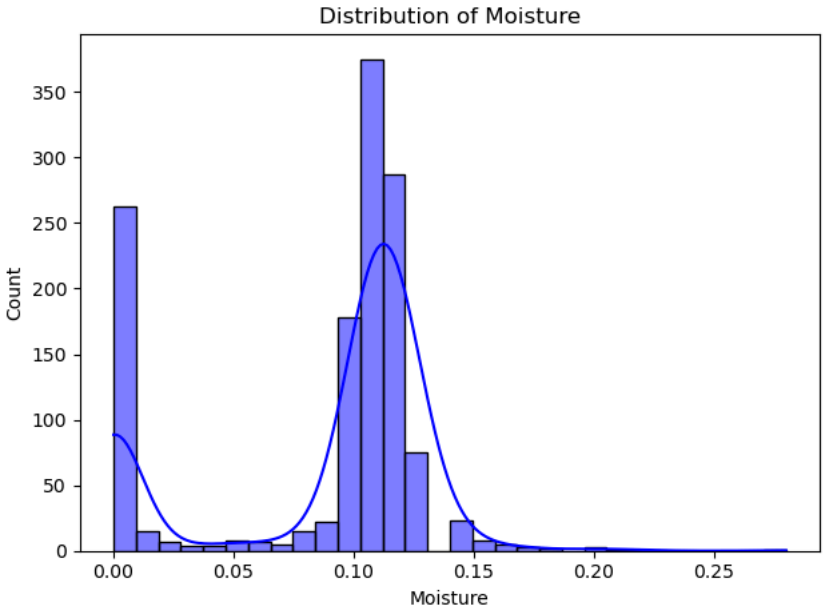

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



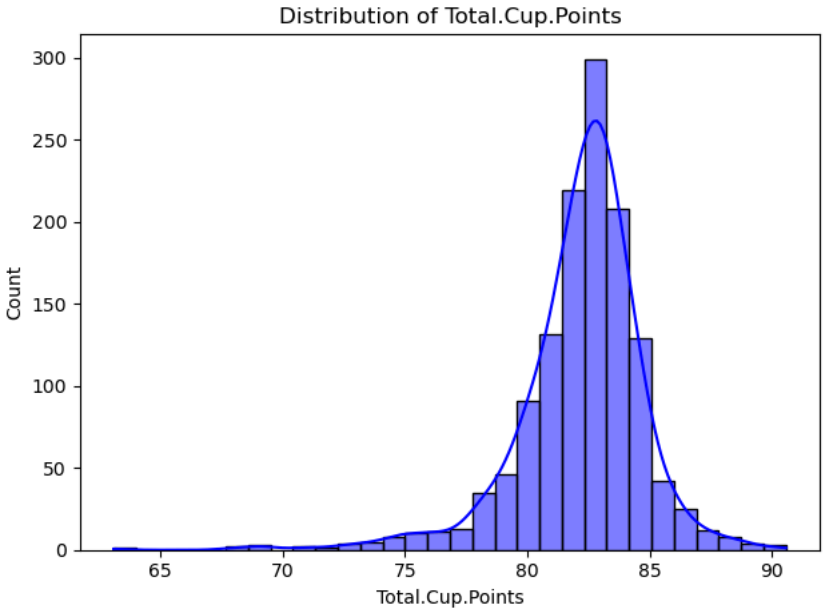

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



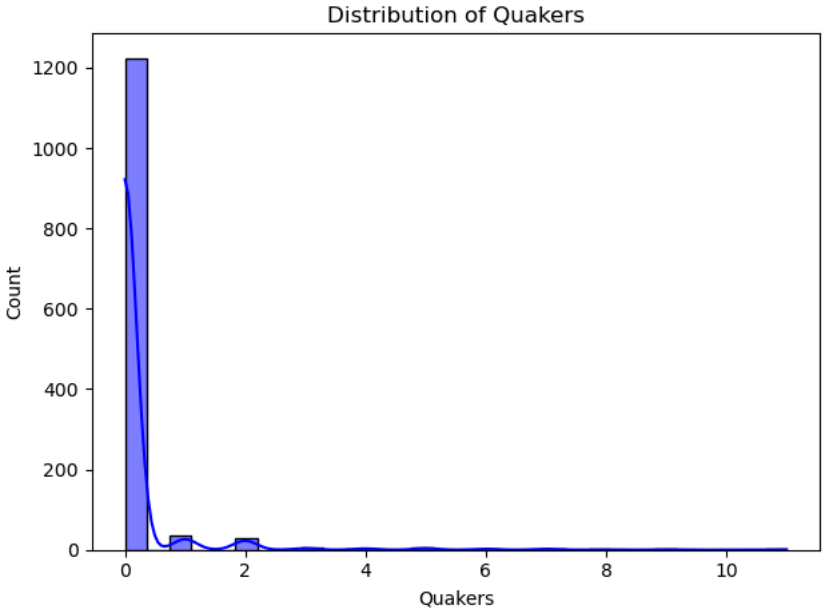

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



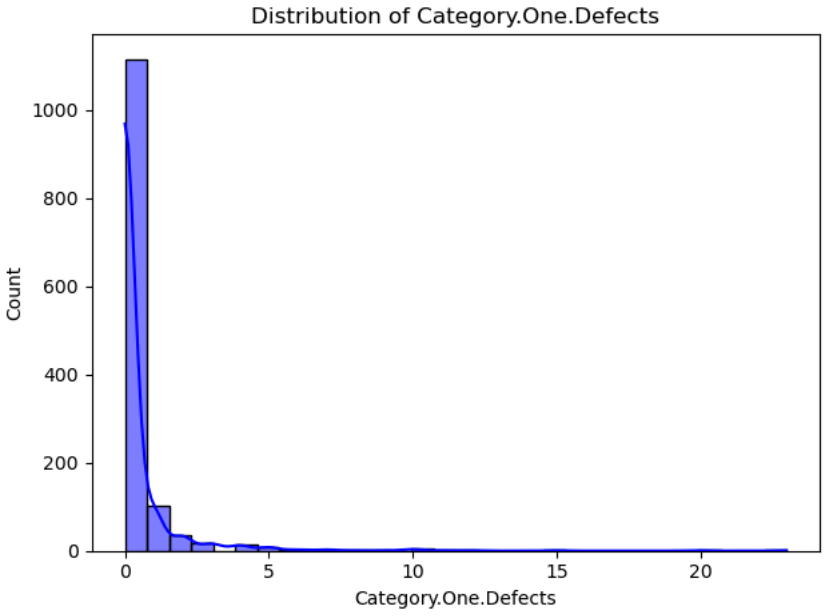

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



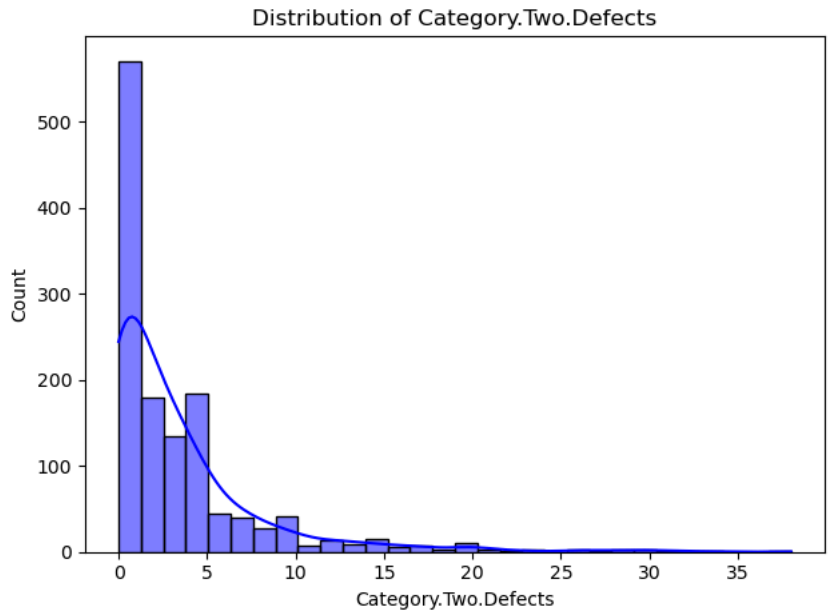

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

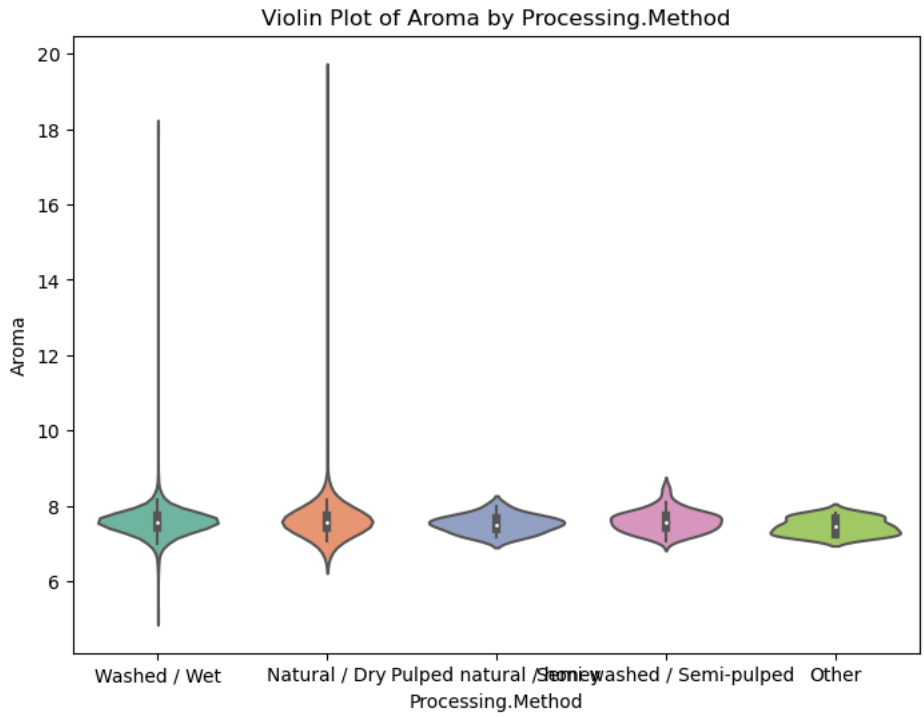


Segmented Univariate Analysis :

- *Analysis of data segments or categories to gain deeper insights.*

In [822]:
```
# Define a function to plot segmented univariate analysis for numerical variables
def plot_segmented_univariate_analysis(df2, numerical_column, segment_column):
    plt.figure(figsize=(18, 6))
    # Violin Plot
    plt.subplot(1, 2, 2)
    sns.violinplot(x=segment_column, y=numerical_column, data=df2, palette='Set2')
    plt.title(f'Violin Plot of {numerical_column} by {segment_column}')
    plt.xlabel(segment_column)
    plt.ylabel(numerical_column)

    plt.show()

# Perform segmented univariate analysis for 'Aroma' across different 'Processing Method'
print("\nSegmented analysis for 'Aroma' by 'Processing Method':")
plot_segmented_univariate_analysis(df2, 'Aroma', 'Processing.Method')
```

```
Segmented analysis for 'Aroma' by 'Processing Method':
```



In [863]:
```
import plotly.express as px
# Check if the required columns are present
required_columns = ['Aroma', 'Flavor', 'Acidity']
if all(column in df2.columns for column in required_columns):
    # Filter the dataframe to include only the required columns and drop any rows with missing values
    df_filtered = df2[required_columns + ['Processing.Method']].dropna()
```

```
    # Create 3D scatter plot
    fig = px.scatter_3d(df_filtered,
                        x='Aroma',
                        y='Flavor',
                        z='Acidity',
                        color='Processing.Method',  # Use the correct column name here
                        title='3D Scatter Plot of Aroma, Flavor, and Acidity by Processing Method',
                        labels={'Aroma': 'Aroma Score', 'Flavor': 'Flavor Score', 'Acidity': 'Acidity Score'})

    # Show the plot
    fig.show()
else:
    print("The required columns are not present in the dataframe.")
```

Insights Generated :

➽ Bivariate Analysis

>> Correlations:

11) Bivariate Analysis :

- *Analysis of relationships between pairs of variables.*

In [867]:
```
corr = quality_measures.corr()
corr
```

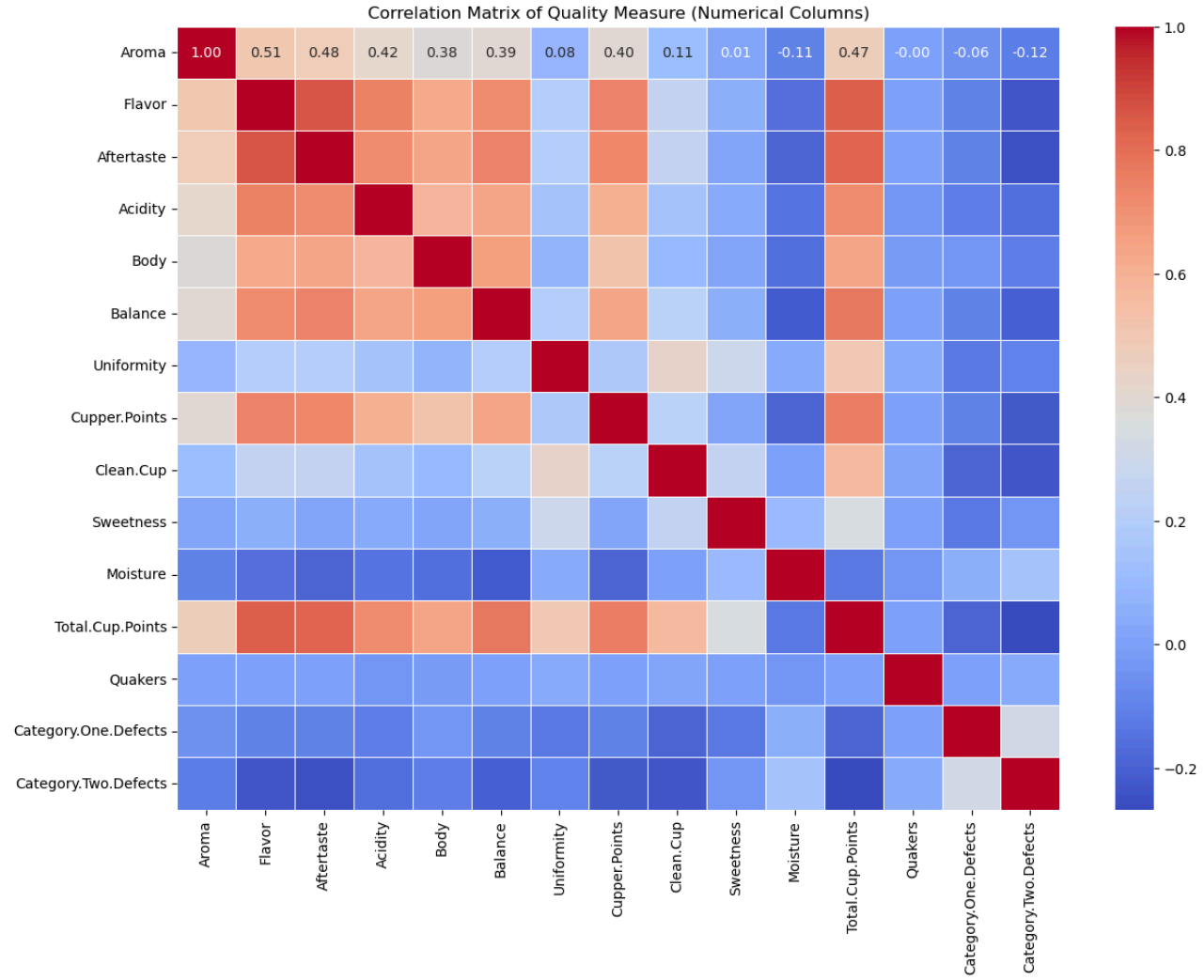
Out [867]:

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Cupper.Points	Clean.Cup	Sweetness	Moist
Aroma	1.000000	0.508439	0.480144	0.416105	0.382436	0.393577	0.084490	0.397754	0.110760	0.011011	-0.105
Flavor	0.508439	1.000000	0.859311	0.745544	0.627433	0.714741	0.201286	0.739345	0.254111	0.050702	-0.156
Aftertaste	0.480144	0.859311	1.000000	0.716186	0.632507	0.742176	0.195325	0.729566	0.251086	0.028843	-0.196
Acidity	0.416105	0.745544	0.716186	1.000000	0.575009	0.630714	0.147355	0.612006	0.135906	0.030904	-0.140
Body	0.382436	0.627433	0.632507	0.575009	1.000000	0.663154	0.071847	0.523495	0.096100	0.019724	-0.164
Balance	0.393577	0.714741	0.742176	0.630714	0.663154	1.000000	0.194029	0.643432	0.222935	0.051131	-0.220
Uniformity	0.084490	0.201286	0.195325	0.147355	0.071847	0.194029	1.000000	0.180527	0.427653	0.298302	0.032
Cupper.Points	0.397754	0.739345	0.729566	0.612006	0.523495	0.643432	0.180527	1.000000	0.226903	0.015596	-0.195
Clean.Cup	0.110760	0.254111	0.251086	0.135906	0.096100	0.222935	0.427653	0.226903	1.000000	0.254107	-0.000
Sweetness	0.011011	0.050702	0.028843	0.030904	0.019724	0.051131	0.298302	0.015596	0.254107	1.000000	0.105
Moisture	-0.105539	-0.156767	-0.196938	-0.140848	-0.164497	-0.220364	0.032522	-0.195375	-0.000214	0.105693	1.0000
Total.Cup.Points	0.473607	0.836144	0.823709	0.715897	0.641770	0.769539	0.499094	0.753961	0.562545	0.338647	-0.138
Quakers	-0.003074	-0.001308	-0.002987	-0.025130	-0.017017	0.004670	0.031915	0.006206	0.027220	0.001681	-0.034
Category.One.Defects	-0.058233	-0.105919	-0.106212	-0.110116	-0.032910	-0.106494	-0.135308	-0.108687	-0.192902	-0.137507	0.047

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Cupper.Points	Clean.Cup	Sweetness	Moist
Category.Two.Defects	-0.121994	-0.231346	-0.243094	-0.167407	-0.111612	-0.204977	-0.100090	-0.224596	-0.228007	-0.040871	0.1342

Heatmap for Correlation of Quality Measures

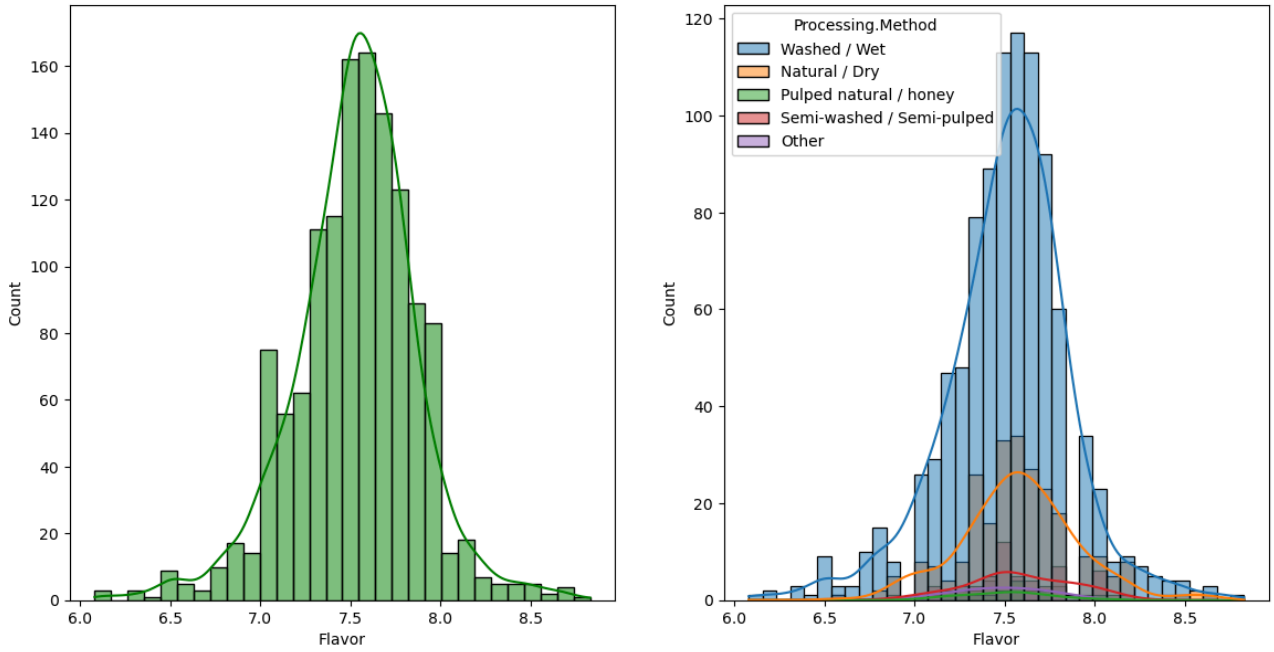
In [870]:
```
# Correlation matrix of quality measures
correlation_matrix = quality_measures.corr()
plt.figure(figsize=(14, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Correlation Matrix of Quality Measure (Numerical Columns)')
plt.show()
```



In [872]:
```
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
plt.subplot(121)
sns.histplot(data=df2,x='Flavor',bins=30,kde =True,color='g')
plt.subplot(122)
sns.histplot(data=df2,x='Flavor',kde =True,hue='Processing.Method')
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.

C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```

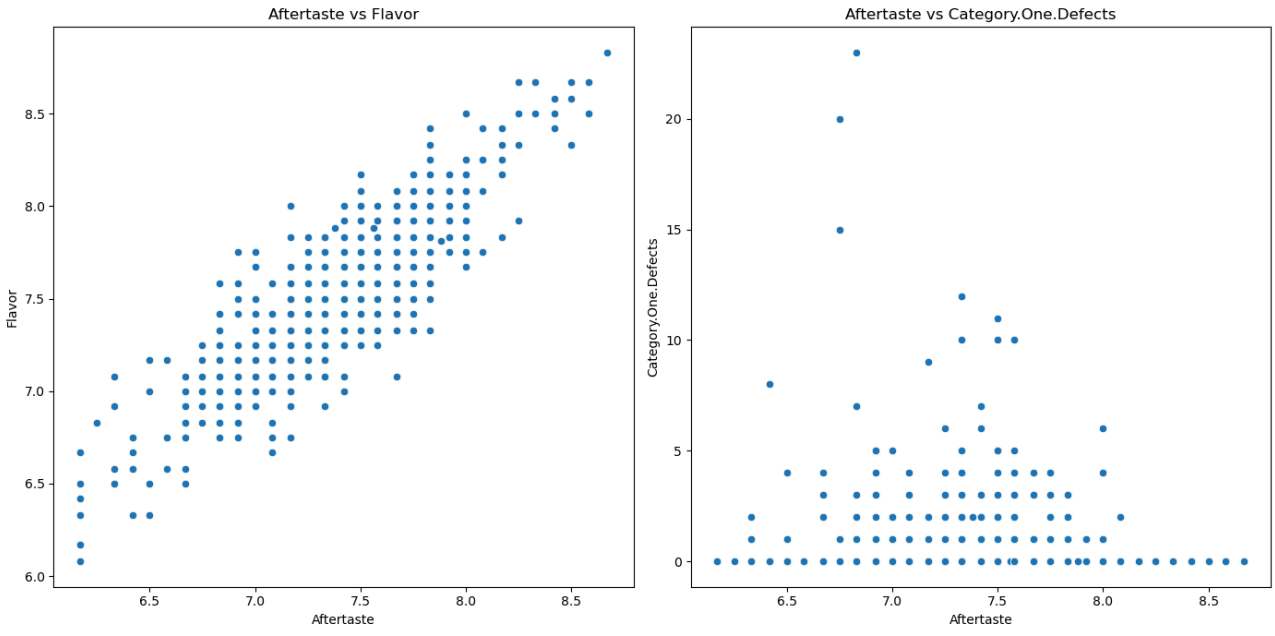
Out [872]: <Axes: xlabel='Flavor', ylabel='Count'>



Comment : From the above graph the Washed/Wet method is used to process coffee beans more than other methods.

>> Scatter Plots:

In [878]:
```
# Scatterplot example: [Aftertaste vs Flavor] and [Aftertaste vs Moisture]
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# ScatterPlot for Aftertaste vs Flavor :
sns.scatterplot(x='Aftertaste', y='Flavor', data=df2, ax=axs[0])
axs[0].set_title('Aftertaste vs Flavor')
# ScatterPlot for Aftertaste vs Category.One.Defects :
sns.scatterplot(x='Aftertaste', y='Category.One.Defects', data=df2, ax=axs[1])
axs[1].set_title('Aftertaste vs Category.One.Defects')
# Display the charts
plt.tight_layout()
plt.show()
```



Results :

- **Positive & Negative Correlation:** The data points form a clear upward trend, indicating a strong positive correlation between "Aftertaste" and "Flavor" whereareas the Aftertaste vs Category.One.Defects is negatively correlated.
- **This means that as the aftertaste score increases**, the flavor score also tends to increase whereaeas the opposite effect on Aftertaste and Category.One.Defects
- **Clustered Points:** Most data points are clustered between aftertaste scores of around 6 to 8 and flavor scores of around 6 to 8. This suggests that the majority of the samples have high scores in both aftertaste and flavor.

>> Chi-Square Test (Chi2):

In [905]:
```
from sklearn.preprocessing import LabelEncoder
from scipy.stats import chi2_contingency
```

```
# Step 1: Encode categorical variables
label = LabelEncoder()
label_encoders = {}

categorical_columns = ['Species', 'Country.of.Origin', 'In.Country.Partner', 'Variety', 'Processing.Method', '
for col in categorical_columns:
    df2[col] = label.fit_transform(df2[col])
    label_encoders[col] = label
```

In [907]:
```
df[categorical_columns].head()
```

Out [907]:

	Species	Country.of.Origin	In.Country.Partner	Variety	Processing.Method	Color
0	0	8	15	5	4	2
1	0	8	15	15	4	2
2	0	9	19	2	4	2
3	0	8	15	5	0	2
4	0	8	15	15	4	2

In [916]:
```
# Perform Chi-Square Test
results = {}

for i in categorical_columns:
    for j in categorical_columns:
        if i != j:
            contingency_table = pd.crosstab(df2[i], df2[j])
            chi2, p, dof, expected = chi2_contingency(contingency_table)
            results[(i, j)] = {'chi2': chi2, 'p-value': p}

# Convert results to a DataFrame for better visualization
chi2_results = pd.DataFrame(results).T
```

In [918]:
```
# Show results
chi2_results
```

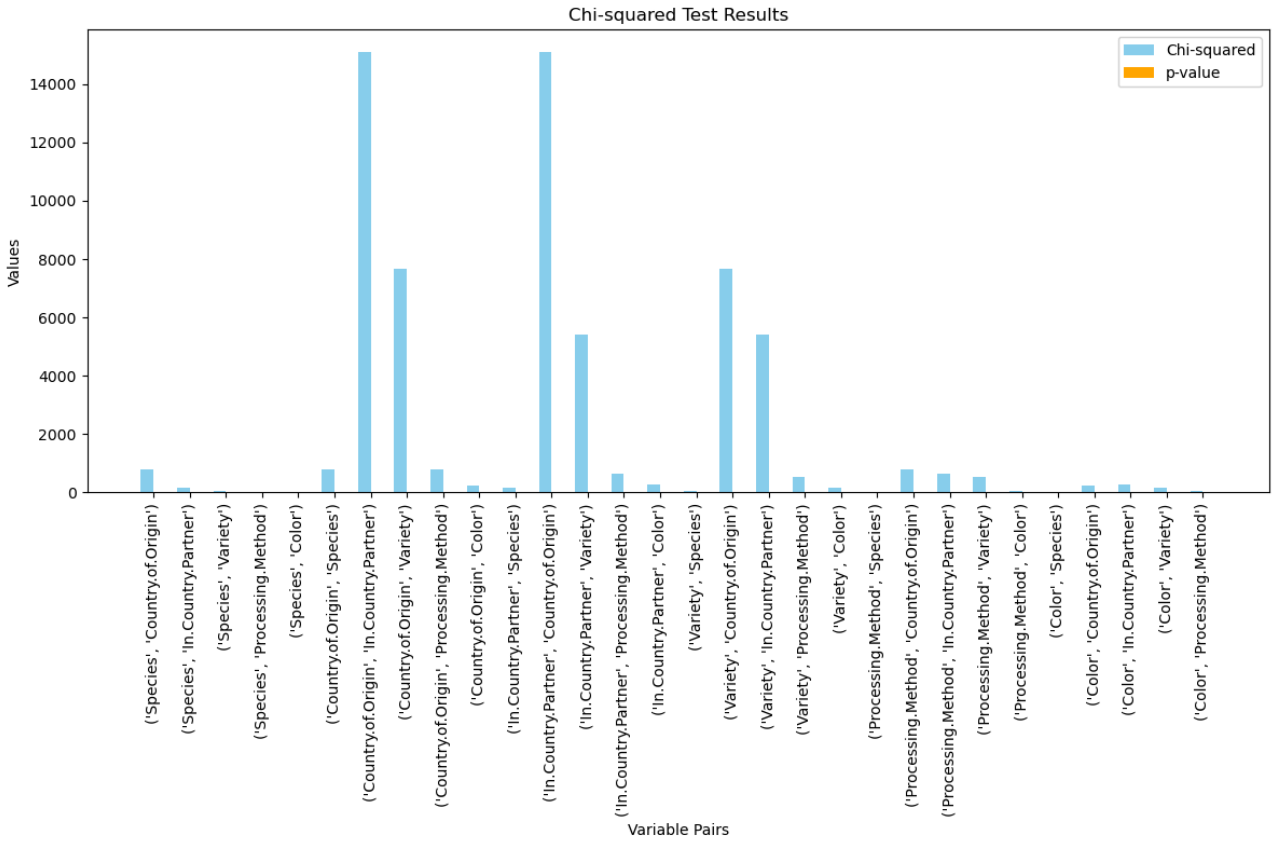
Out [918]:

		chi2	p-value
Species	**Country.of.Origin**	799.089834	3.099531e-145
	In.Country.Partner	178.878937	3.328648e-25
	Variety	43.084575	3.415564e-02
	Processing.Method	2.232002	6.931748e-01
	Color	0.990779	6.093336e-01
Country.of.Origin	**Species**	799.089834	3.099531e-145
	In.Country.Partner	15106.812537	0.000000e+00
	Variety	7651.534612	0.000000e+00
	Processing.Method	793.845899	6.006653e-92
	Color	226.846871	1.918919e-18
In.Country.Partner	**Species**	178.878937	3.328648e-25
	Country.of.Origin	15106.812537	0.000000e+00
	Variety	5396.600990	0.000000e+00
	Processing.Method	638.597458	2.108365e-79
	Color	267.143252	1.996102e-31
Variety	**Species**	43.084575	3.415564e-02
	Country.of.Origin	7651.534612	0.000000e+00
	In.Country.Partner	5396.600990	0.000000e+00
	Processing.Method	537.639406	7.390320e-57
	Color	156.941544	1.658389e-11
Processing.Method	**Species**	2.232002	6.931748e-01
	Country.of.Origin	793.845899	6.006653e-92
	In.Country.Partner	638.597458	2.108365e-79
	Variety	537.639406	7.390320e-57
	Color	44.417880	4.743247e-07
Color	**Species**	0.990779	6.093336e-01
	Country.of.Origin	226.846871	1.918919e-18
	In.Country.Partner	267.143252	1.996102e-31
	Variety	156.941544	1.658389e-11

	chi2	p-value
Processing.Method	44.417880	4.743247e-07

Charts for Chi-Square Test :

In [1003]:
```
# Extract data for plotting
pairs = chi2_results.index
chi2_values = chi2_results['chi2']
p_values = chi2_results['p-value']

# Set the positions for the bars
positions = range(len(pairs))

# Plotting the grouped bar chart
plt.figure(figsize=(12, 8))
bar_width = 0.35
plt.bar(positions, chi2_values, bar_width, label='Chi-squared', color='skyblue')
plt.bar([p + bar_width for p in positions], p_values, bar_width, label='p-value', color='orange')

# Add labels and title
plt.xlabel('Variable Pairs')
plt.ylabel('Values')
plt.title('Chi-squared Test Results')
plt.xticks([p + bar_width/2 for p in positions], pairs, rotation=90)
plt.legend()

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



In [1011]:
```
chi2_results.index = pd.MultiIndex.from_tuples(chi2_results.index, names=['Variable 1', 'Variable 2'])

# Create a pivot table for the heatmap
chi2_pivot = chi2_results.reset_index().pivot(index='Variable 1', columns='Variable 2', values='p-value')

# Create a heatmap to visualize the p-values
plt.figure(figsize=(14, 10))
sns.heatmap(chi2_pivot, annot=True, cmap='viridis', fmt=".4f", cbar_kws={'label': 'p-value'})
plt.title('Heatmap of Chi-Square Test p-values')
plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\matrix.py:260: FutureWarning:

Format strings passed to MaskedConstant are ignored, but in future may error or produce different behavior
```

Key Insights:

✨ **Significant Associations:**

- **Species and Country of Origin**: The chi-squared statistic is 799.09 with a very low p-value (3.099531e-145), indicating a very strong association between coffee species and country of origin.
- **In Country Partner and Country of Origin**: The chi-squared statistic is 15106.81 with a p-value close to zero, indicating a strong association between in-country partner and country of origin.
- **Variety and Country of Origin**: The chi-squared statistic is 7651.53 with a p-value of zero, indicating a strong association between coffee variety and country of origin.
- **Processing Method and Country of Origin**: The chi-squared statistic is 793.85 with a p-value (6.006653e-92), showing a significant association between processing method and country of origin.
- **Color and Country of Origin**: The chi-squared statistic is 226.85 with a p-value (1.918919e-18), indicating a significant association between coffee color and country of origin.

✨ **Moderate Associations:**

- **Species and In Country Partner**: The chi-squared statistic is 178.88 with a p-value (3.328648e-25), suggesting a significant, though not as strong, association between species and in-country partner.
- **Processing Method and In Country Partner**: The chi-squared statistic is 638.60 with a p-value (2.108365e-79), indicating a significant association.
- **Variety and In Country Partner**: The chi-squared statistic is 5396.60 with a p-value of zero, showing a strong association.

✨ **Weak or No Associations:**

- **Processing Method and Species**: The chi-squared statistic is 2.23 with a high p-value (6.931748e-01), suggesting no significant association.
- **Color and Species**: The chi-squared statistic is 0.99 with a p-value (6.093336e-01), indicating no significant association.
- **Color and Processing Method**: The chi-squared statistic is 44.42 with a p-value (4.743247e-07), which is significant but not as strong compared to other associations.

🚀 **Analysed Relationships/ Intrepretation:**

- **Strongest Associations**: The Country of Origin appears to be a key factor influencing various other attributes of coffee, such as species, in-country partner, variety, processing method, and color.
- **Moderate Associations**: Attributes like species and in-country partner have notable associations, which might be relevant for understanding regional coffee characteristics.
- **Weak Associations**: There is little to no significant relationship between processing method and species, as well as color and species, suggesting these attributes may vary independently of each other.

>> ANOVA (F_classif):

ANOVA Feature Classif : (NUM to CAT)

- *Selecting Total.Cup.Points as target (y) beacause it seems like a strong candidate for the target variable as it directly indicates the overall quality of the coffee.*

- *And also comparing some other numerical features with categorical features.*

In [942]:
```
x1 = df2[categorical_columns]
y = df2['Total.Cup.Points']
y1 = df2['Flavor']
y2 = df2['Balance']
y3 = df2['Acidity']
```

In [944]:
```
from sklearn.feature_selection import SelectKBest, f_classif
best_features = SelectKBest(score_func=f_classif, k=2)
fit = best_features.fit(x1,y)
selected_features = x1.columns[fit.get_support()]
print("Selected Features for Total.Cup.Points:", selected_features)
```

```
Selected Features for Total.Cup.Points: Index(['In.Country.Partner', 'Processing.Method'], dtype='object')
```

In [946]:
```
from sklearn.feature_selection import SelectKBest, f_classif
best_features = SelectKBest(score_func=f_classif, k=2)
fit = best_features.fit(x1,y1)
selected_features = x1.columns[fit.get_support()]
print("Selected Features for Flavor:", selected_features)
```

```
Selected Features for Flavor: Index(['Country.of.Origin', 'In.Country.Partner'], dtype='object')
```

In [948]:
```
from sklearn.feature_selection import SelectKBest, f_classif
best_features = SelectKBest(score_func=f_classif, k=2)
fit = best_features.fit(x1,y2)
selected_features = x1.columns[fit.get_support()]
print("Selected Features for Balance:", selected_features)
```

```
Selected Features for Balance: Index(['Species', 'In.Country.Partner'], dtype='object')
```

In [950]:
```
from sklearn.feature_selection import SelectKBest, f_classif
best_features = SelectKBest(score_func=f_classif, k=2)
fit = best_features.fit(x1,y3)
selected_features = x1.columns[fit.get_support()]
print("Selected Features of Acidity:", selected_features)
```

```
Selected Features of Acidity: Index(['In.Country.Partner', 'Variety'], dtype='object')
```

Insights from Selected Features for Coffee Quality Measures

Total.Cup.Points:

Selected Features:

- `In.Country.Partner`
- `Processing.Method`

Insights:

- **In.Country.Partner**: The relationship with the in-country partner, likely referring to the local entity or organization involved in the coffee's production or quality assessment, plays a significant role in the overall cup score. This could be due to the varying standards and practices of different partners in processing and evaluating coffee.
- **Processing.Method**: The method used to process the coffee beans (e.g., washed, natural, honey) has a significant impact on the total cup points. This is expected as the processing method affects the flavor profile, cleanliness, and other sensory attributes of the coffee.

Flavor:

Selected Features:

- `Country.of.Origin`
- `In.Country.Partner`

Insights:

- **Country.of.Origin**: The origin country of the coffee is a critical determinant of its flavor. Different countries, due to their unique climates, soil types, and coffee varieties, produce beans with distinctive flavor profiles.
- **In.Country.Partner**: The local partner's influence also plays a significant role in the flavor, suggesting that local practices and expertise in handling and processing coffee beans can significantly alter the flavor profile.

Balance:

Selected Features:

- `Species`
- `In.Country.Partner`

Insights:

- **Species**: The coffee species (e.g., Arabica, Robusta) is crucial in determining the balance of the coffee. Different species have inherent characteristics that influence how balanced the flavor profile is.
- **In.Country.Partner**: Again, the involvement of the local partner is significant, indicating that their practices in growing, harvesting, and processing the beans can affect the balance of the coffee.

Acidity:

Selected Features:

- In.Country.Partner
- Variety

Insights:

- **In.Country.Partner**: The local partner's impact on acidity is noteworthy, suggesting that local methods and environmental factors managed by the partner affect the acidity levels in coffee.
- **Variety**: The specific variety of coffee (e.g., Bourbon, Typica) influences the acidity. Different varieties have genetic differences that result in varying acidity levels in the cup.

General Insights:

- **In.Country.Partner**: This feature appears consistently across all quality measures, highlighting the significant role of local partners in influencing coffee quality. This could encompass their expertise, quality control measures, and specific practices in coffee handling and processing.
- **Processing.Method**: Specifically influential for the total cup points, indicating the critical impact of how the coffee is processed after harvest.
- **Country.of.Origin**: Plays a crucial role in flavor, emphasizing the importance of geographical and climatic conditions in shaping the sensory characteristics of coffee.
- **Species and Variety**: These genetic factors are essential for specific attributes like balance and acidity, underscoring the intrinsic qualities of the coffee plants themselves.

These insights can guide stakeholders in the coffee industry, such as farmers, processors, and marketers, to focus on these critical factors to enhance coffee quality. Understanding these relationships can help in making informed decisions about cultivation practices, processing methods, and partnership selections to achieve desired quality outcomes.

In [961]:
```
numerical_columns = df2[['Number.of.Bags', 'Bag.Weight', 'Aroma', 'Flavor', 'Aftertaste',
       'Acidity', 'Body', 'Balance', 'Uniformity', 'Clean.Cup', 'Sweetness',
       'Cupper.Points', 'Total.Cup.Points', 'Moisture', 'Category.One.Defects',
       'Quakers', 'Category.Two.Defects']]
```

AUTO ANOVA Feature Classif : (CAT to NUM)

- *Now Comparing the Categorical Features with Numerical Features of 'Country.of.Origin' and 'Processing.Method'.*

In [964]:
```
# For Country.of.Origin :
x2 = numerical_columns
y4 = df2['Country.of.Origin']
# Initialize SelectKBest with f_classif score function
best_features = SelectKBest(score_func=f_classif, k=2)
fit = best_features.fit(x2, y4)
selected_features = x2.columns[fit.get_support()]
print("Selected Features:", selected_features)

anova_stats, anova_p_values = f_classif(x2, y4)
anova_results_cat = pd.DataFrame({'Feature': x2.columns, 'F Stat': anova_stats, 'P-value': anova_p_values})
```

```
Selected Features: Index(['Number.of.Bags', 'Sweetness'], dtype='object')
```

In [966]:
```
anova_results_cat
```

Out [966]:

	Feature	F Stat	P-value
0	Number.of.Bags	25.480386	1.764103e-121
1	Bag.Weight	1.523371	2.662345e-02
2	Aroma	5.272383	1.899711e-20
3	Flavor	9.046545	3.158334e-41
4	Aftertaste	11.667931	2.461880e-55
5	Acidity	10.183243	2.131956e-47
6	Body	11.667481	2.475422e-55
7	Balance	11.983125	5.317317e-57
8	Uniformity	3.104323	5.206005e-09
9	Clean.Cup	2.417370	9.029156e-06
10	Sweetness	16.249817	8.014149e-79
11	Cupper.Points	7.766437	3.352928e-34
12	Total.Cup.Points	8.774658	9.678481e-40
13	Moisture	11.415124	5.398175e-54
14	Category.One.Defects	2.628322	9.938988e-07
15	Quakers	1.090843	3.300645e-01
16	Category.Two.Defects	5.590570	3.421925e-22

In [968]:
```
# For Processing.Method :
x2 = numerical_columns
y5 = df2['Processing.Method']
# Initialize SelectKBest with f_classif score function
best_features = SelectKBest(score_func=f_classif, k=2)
fit = best_features.fit(x2, y5)
selected_features = x2.columns[fit.get_support()]
print("Selected Features:", selected_features)
```

```
anova_stats, anova_p_values = f_classif(x2, y5)
anova_results_cat1 = pd.DataFrame({'Feature': x2.columns, 'F Stat': anova_stats, 'P-value': anova_p_values})
```

```
Selected Features: Index(['Body', 'Quakers'], dtype='object')
```

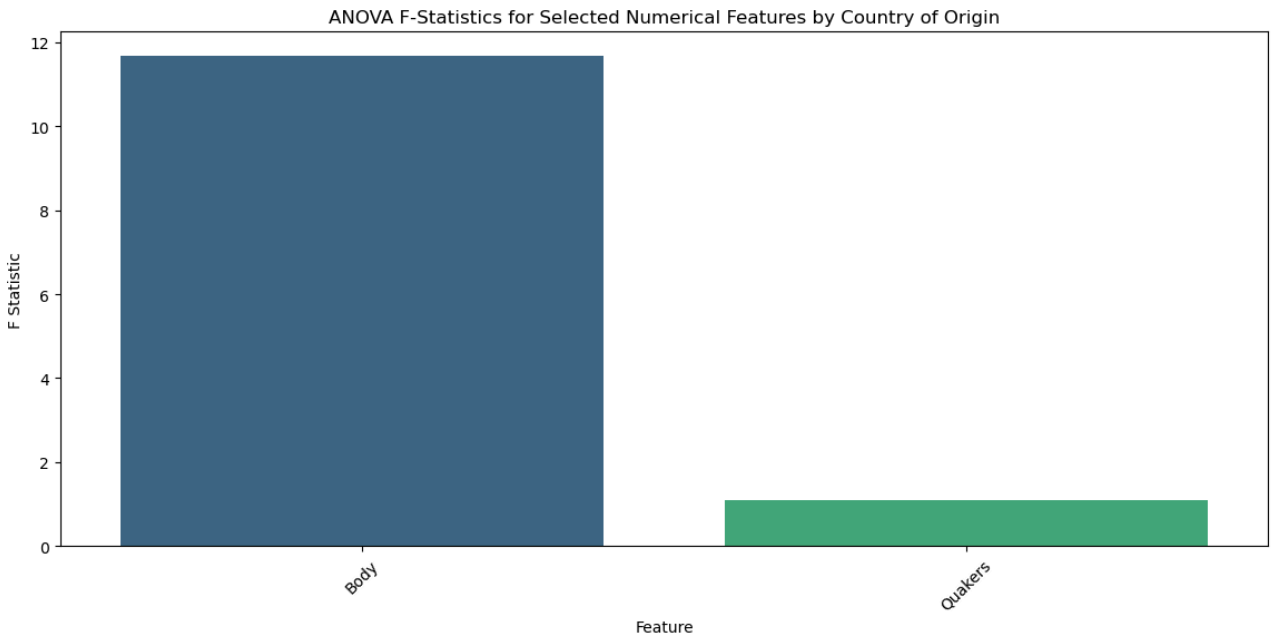
In [970]:
```
anova_results_cat1
```

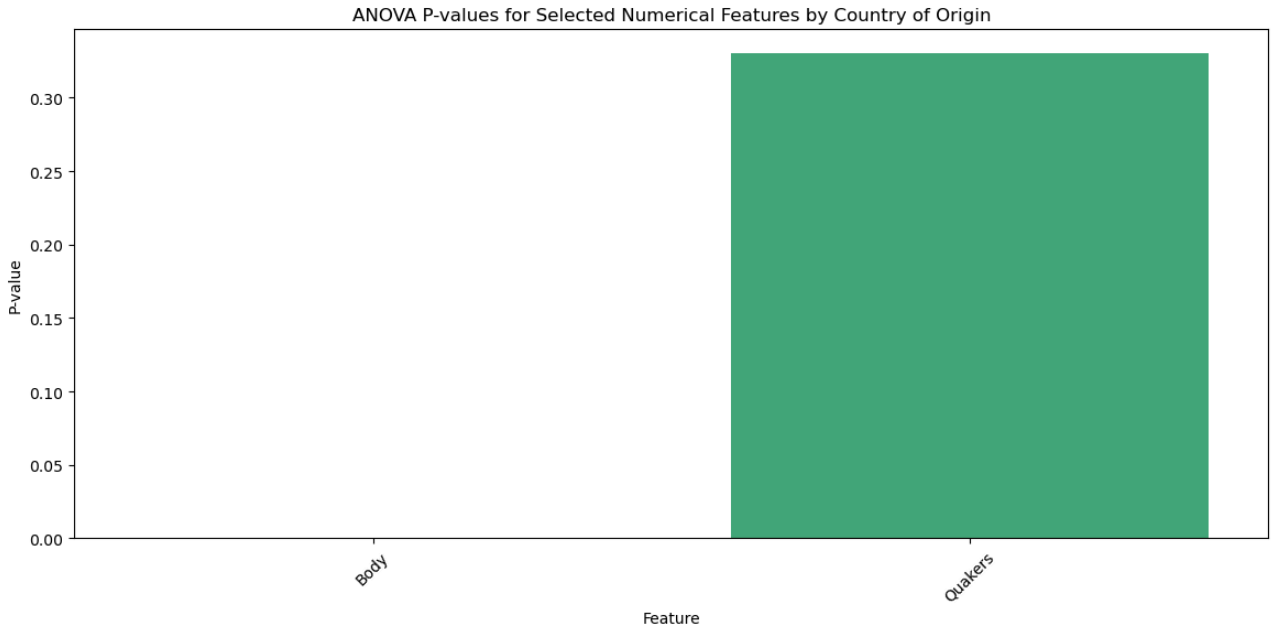
Out [970]:

	Feature	F Stat	P-value
0	Number.of.Bags	3.374498	9.318407e-03
1	Bag.Weight	0.152869	9.617434e-01
2	Aroma	1.023394	3.939225e-01
3	Flavor	2.837926	2.327918e-02
4	Aftertaste	2.772290	2.599877e-02
5	Acidity	1.264402	2.820331e-01
6	Body	5.429583	2.455748e-04
7	Balance	2.995515	1.782911e-02
8	Uniformity	1.243548	2.905402e-01
9	Clean.Cup	0.815089	5.154887e-01
10	Sweetness	4.340513	1.719830e-03
11	Cupper.Points	3.395521	8.986515e-03
12	Total.Cup.Points	1.648836	1.595869e-01
13	Moisture	3.080156	1.543726e-02
14	Category.One.Defects	2.273576	5.937978e-02
15	Quakers	15.396516	2.601653e-12
16	Category.Two.Defects	0.493770	7.403395e-01

Charts for ANOVA F-Test :

In [1018]:
```
# Visualize the selected features' ANOVA F-statistics and p-values
selected_anova_results = anova_results_cat[anova_results_cat['Feature'].isin(selected_features)]

# Plot F-statistics
plt.figure(figsize=(14, 6))
sns.barplot(x='Feature', y='F Stat', data=selected_anova_results, palette='viridis')
plt.title('ANOVA F-Statistics for Selected Numerical Features by Country of Origin')
plt.xlabel('Feature')
plt.ylabel('F Statistic')
plt.xticks(rotation=45)
plt.show()

# Plot p-values
plt.figure(figsize=(14, 6))
sns.barplot(x='Feature', y='P-value', data=selected_anova_results, palette='viridis')
plt.title('ANOVA P-values for Selected Numerical Features by Country of Origin')
plt.xlabel('Feature')
plt.ylabel('P-value')
plt.xticks(rotation=45)
plt.show()
```





General Insights :

Selected Features for Coffee Quality Measures

- **Number.of.Bags & Sweetness**:
  - **Number.of.Bags**: Highly significant (F Stat: 25.480386, P-value: 1.764103e-121) influencing quality measures.
  - **Sweetness**: Very significant (F Stat: 16.249817, P-value: 8.014149e-79) affecting various quality attributes.

Top ANOVA Results for Coffee Quality

- **Balance**: Highly significant (F Stat: 11.983125, P-value: 5.317317e-57).
- **Aftertaste**: Strong impact (F Stat: 11.667931, P-value: 2.461880e-55).
- **Body**: Significant (F Stat: 11.667481, P-value: 2.475422e-55).
- **Sweetness**: Notable influence (F Stat: 16.249817, P-value: 8.014149e-79).

Selected Features for Processing Method

- **Body & Quakers**:
  - **Body**: Significant influence (F Stat: 5.429583, P-value: 2.455748e-04).
  - **Quakers**: Highly significant (F Stat: 15.396516, P-value: 2.601653e-12).

Summary

- **Key Influencers**: Number of Bags, Sweetness, Balance, Aftertaste, and Body are major factors affecting coffee quality.
- **Processing Method**: Body and Quakers are crucial for determining the processing method's impact.

➽ Multivariate Analysis

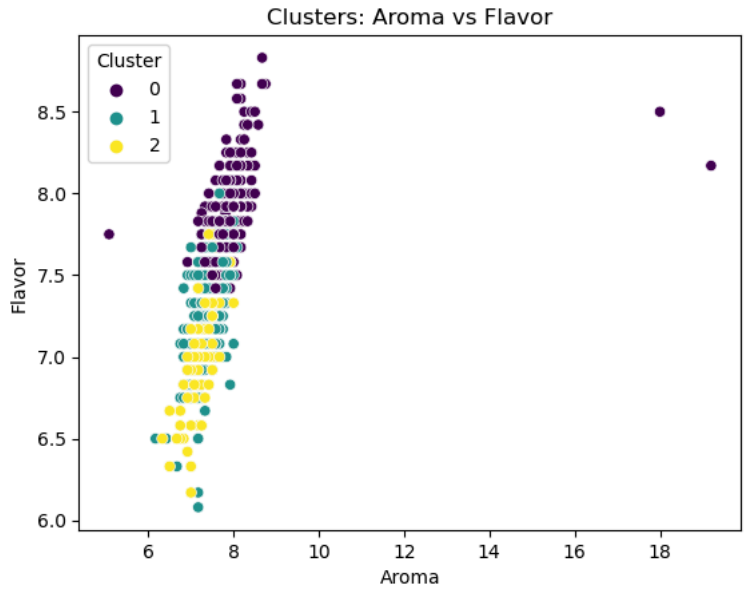
- *Explore interactions and dependencies among multiple variables using techniques like clustering or dimensionality reduction.*

>> Clustering:

In [979]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

```
# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(quality_measures)
# KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df2['Cluster'] = kmeans.fit_predict(data_scaled)
# Plot clusters
plt.figure()
sns.scatterplot(x='Aroma', y='Flavor', hue='Cluster', data=df2, palette='viridis')
plt.title('Clusters: Aroma vs Flavor')
plt.show()
```

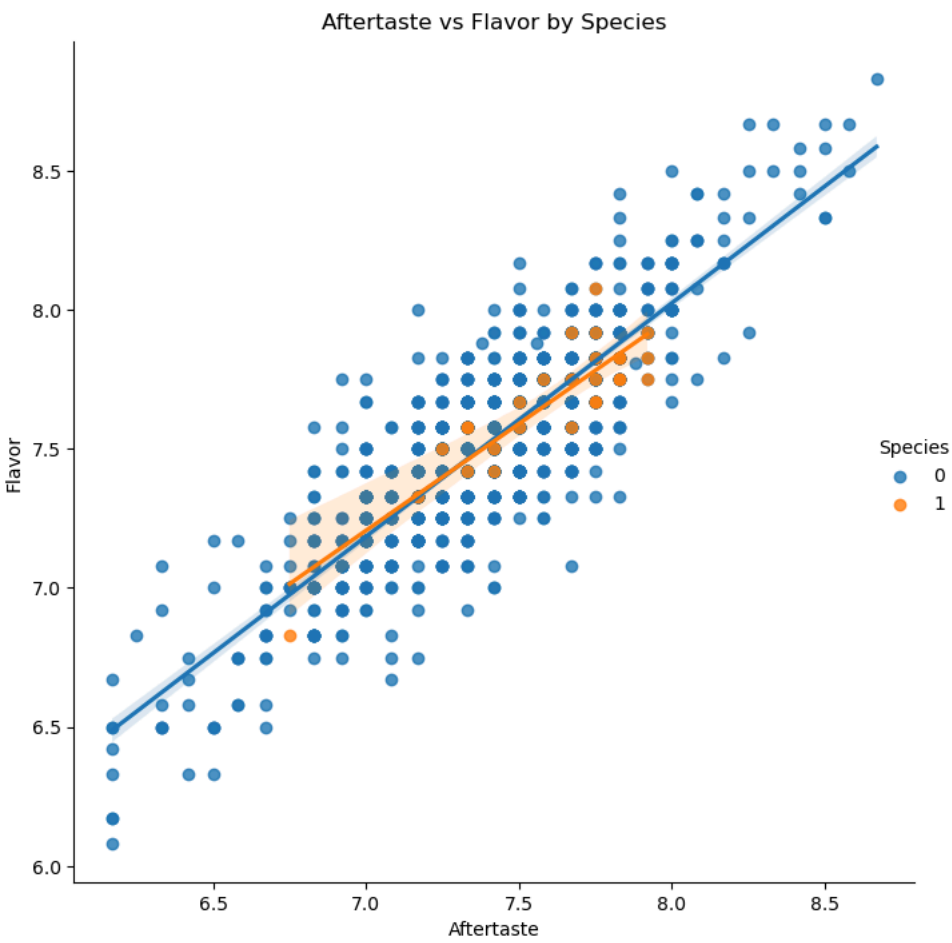
```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

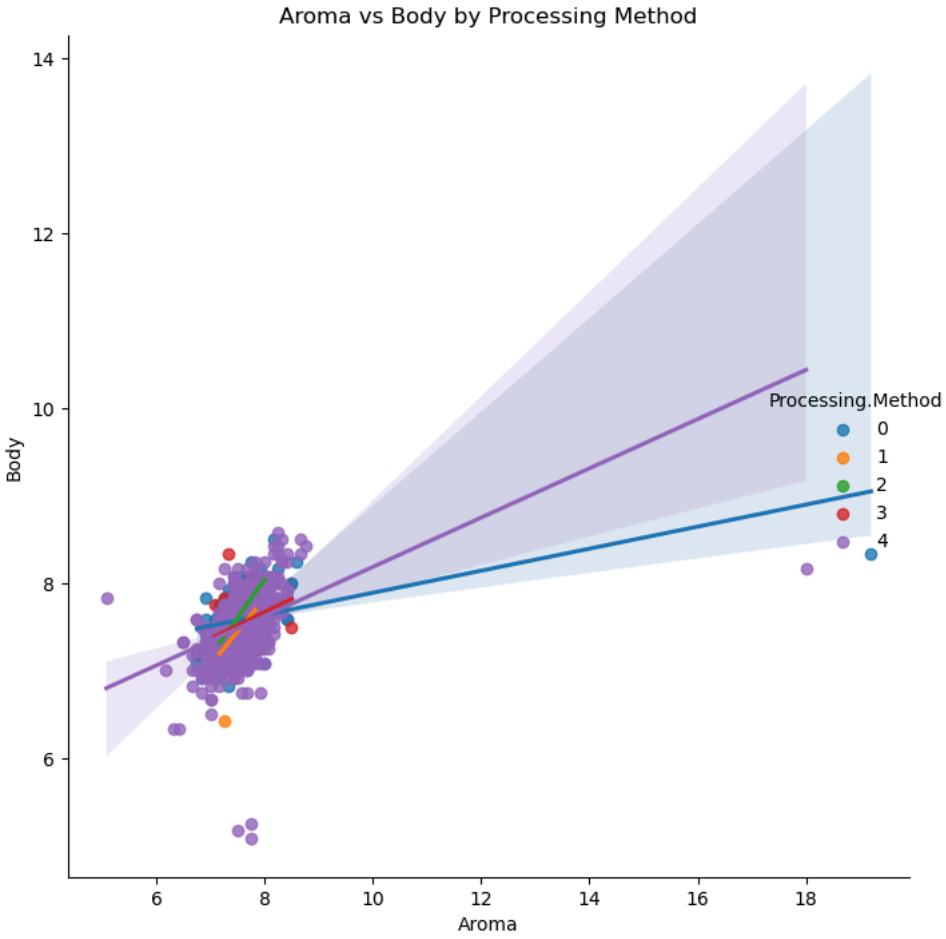


Some other comparison in lmplot :

In [982]:

```
# Create the first lmplot and adjust its position
g1 = sns.lmplot(x='Aftertaste', y='Flavor', hue='Species', data=df2, aspect=1.5)
g1.set_axis_labels('Aftertaste', 'Flavor')
g1.fig.set_size_inches(7, 7)
g1.fig.subplots_adjust(left=0.05, right=0.95, top=0.95, bottom=0.05)
g1.fig.suptitle('Aftertaste vs Flavor by Species')

# Create the second lmplot and adjust its position
g2 = sns.lmplot(x='Aroma', y='Body', hue='Processing.Method', data=df2, aspect=1.5)
g2.set_axis_labels('Aroma', 'Body')
g2.fig.set_size_inches(7, 7)
g2.fig.subplots_adjust(left=0.05, right=0.95, top=0.95, bottom=0.05)
g2.fig.suptitle('Aroma vs Body by Processing Method')

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





Multivariate Analysis using Pairplot

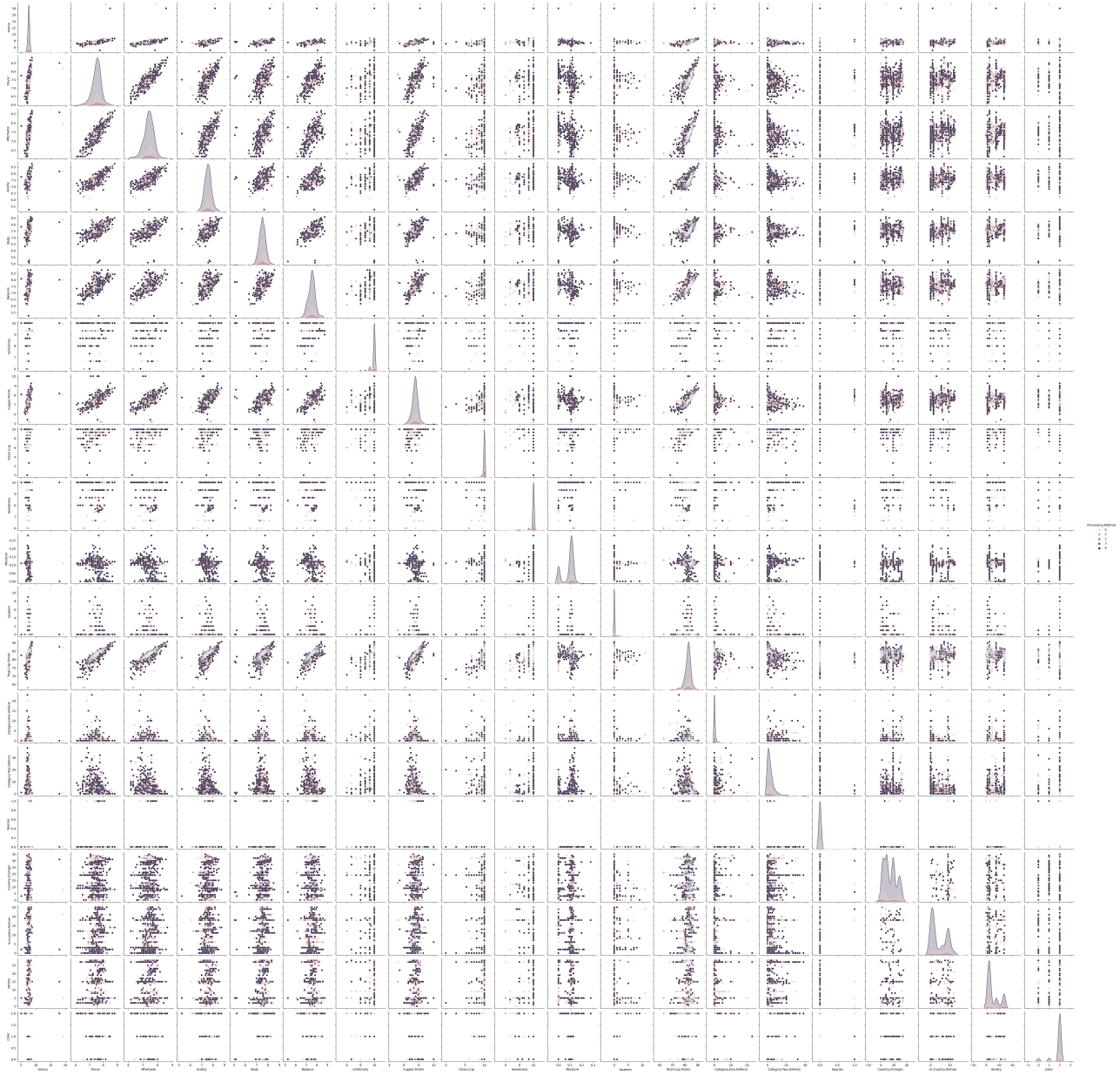
In [987]:

```
pairplot_chart = df2[['Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance',
       'Uniformity', 'Cupper.Points', 'Clean.Cup', 'Sweetness', 'Moisture',
       'Quakers', 'Total.Cup.Points', 'Category.One.Defects',
       'Category.Two.Defects','Species', 'Country.of.Origin', 'In.Country.Partner', 'Variety',
       'Processing.Method', 'Color']]
```

In [989]:

```python
sns.pairplot(pairplot_chart,hue= 'Processing.Method')
plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
C:\Users\Admin\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```



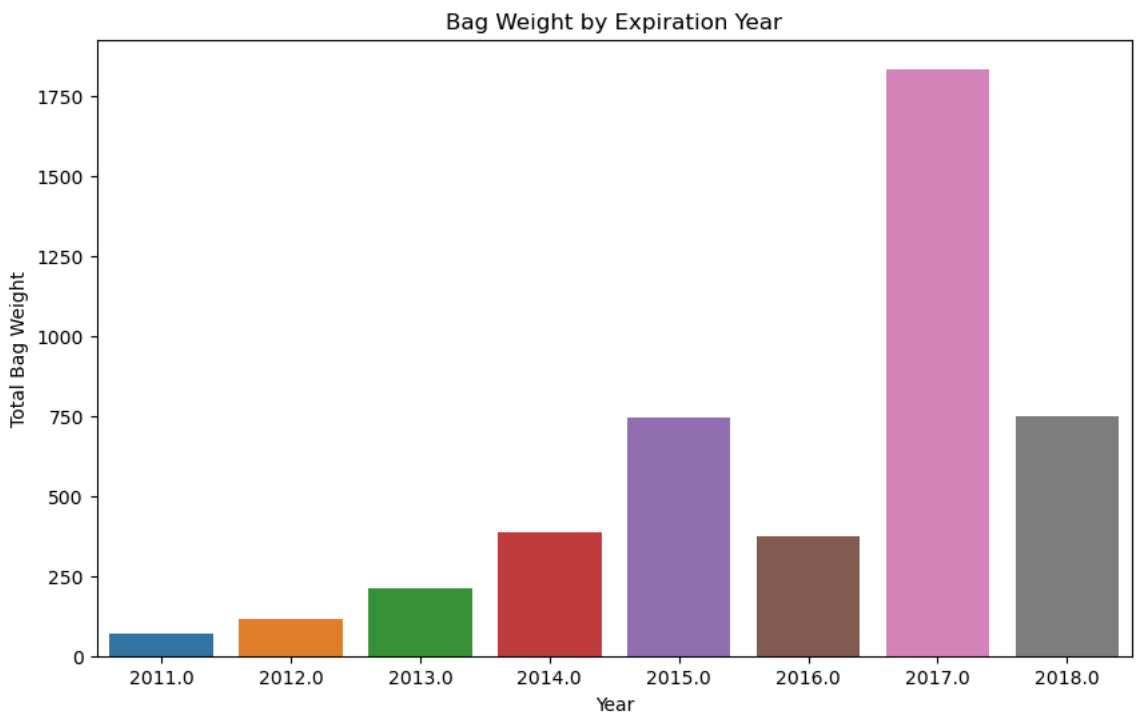
III) Visualization:

- Utilize appropriate visualization techniques (histograms, boxplots, scatterplots, etc.) to visually represent the distribution and relationships within the dataset.
- Create insightful visualizations to highlight patterns, trends, and anomalies in the data, facilitating better understanding and interpretation.

Combination Viz 1 : *Let's Combine* ***"Bag.Weight"*** *and* ***"Expiration Year"*** *for some visualization.*

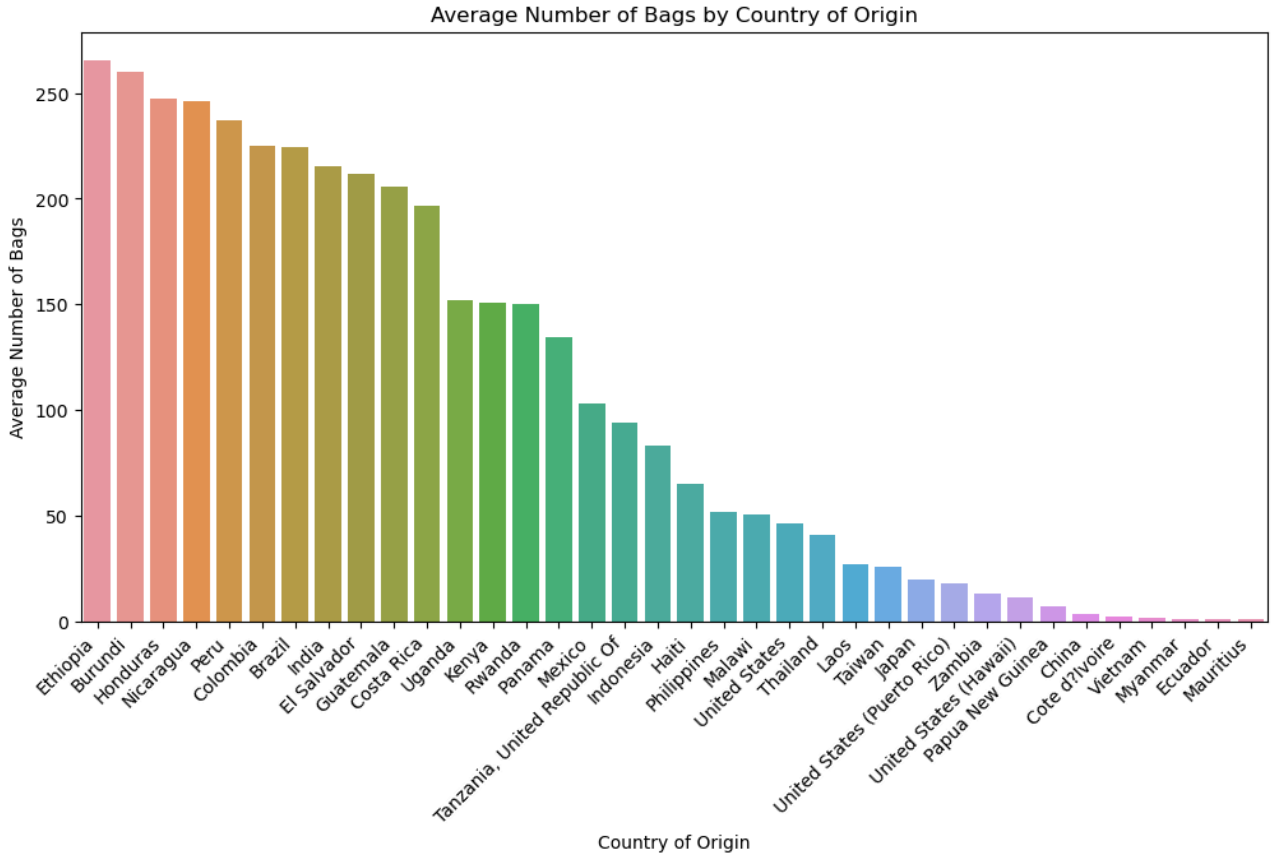
In [1304]:

```
# Bag Weight by Expiration Year
df2['Expiration'] = pd.to_datetime(df2['Expiration'], errors='coerce')

df2['Year'] = df2['Expiration'].dt.year
df_yearly = df2.groupby('Year').agg({'Bag.Weight': 'sum'}).reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Year', y='Bag.Weight', data=df_yearly)
plt.title('Bag Weight by Expiration Year')
plt.xlabel('Year')
plt.ylabel('Total Bag Weight')
plt.show()
```



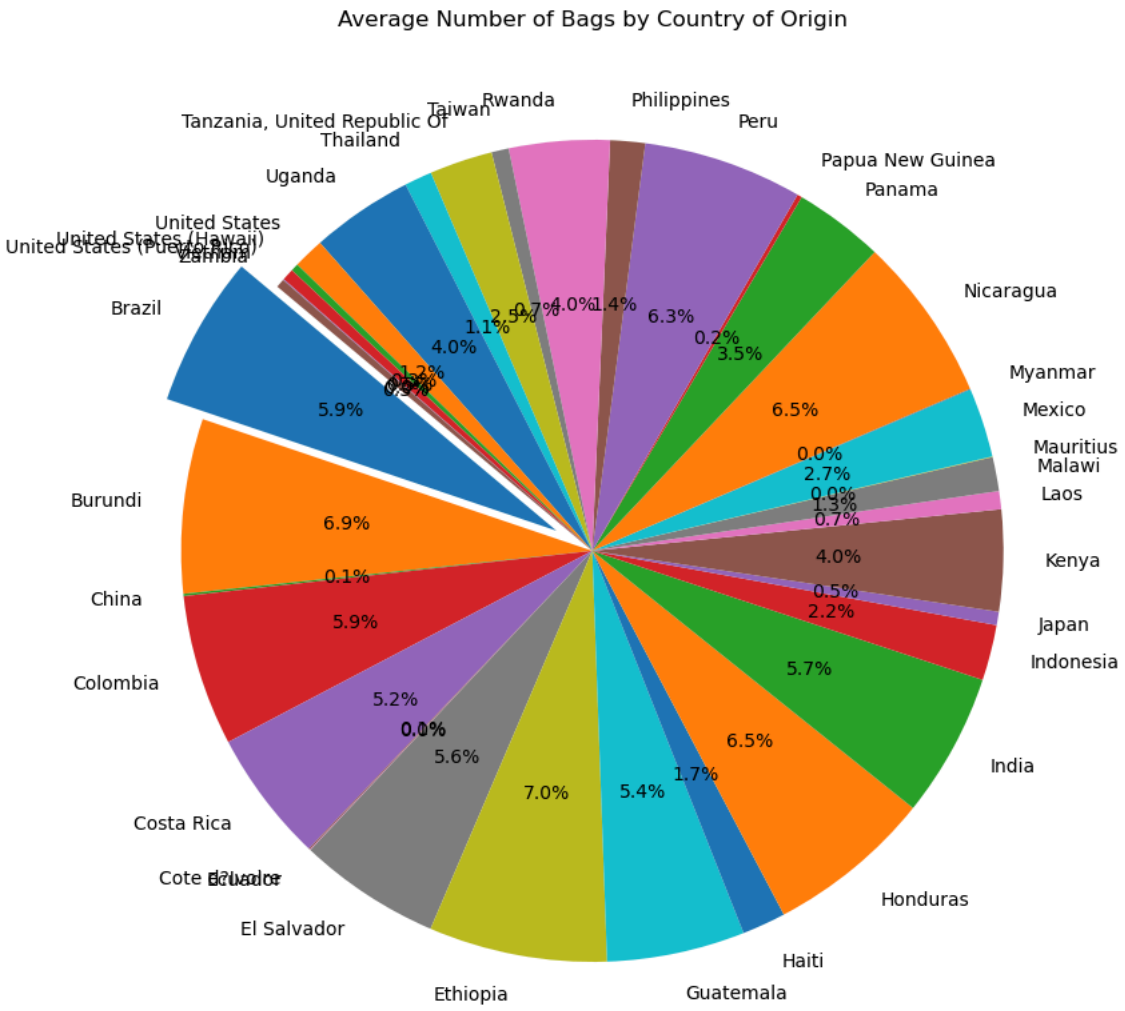
Combination Viz 2 : *Let's Combine "**Country.of.Origin**" and "**Number>of.Bags**" for some visualization.*

In [1213]:
```
# Average Number of Bags by Country of Origin
average_bags_per_origin = df4.groupby('Country.of.Origin')['Number.of.Bags'].mean().reset_index()

sorted_countries = average_bags_per_origin.sort_values(by='Number.of.Bags', ascending=False)['Country.of.Origi

plt.figure(figsize=(12, 6))
sns.barplot(x='Country.of.Origin', y='Number.of.Bags', data=average_bags_per_origin, order=sorted_countries, e
plt.title('Average Number of Bags by Country of Origin')
plt.xlabel('Country of Origin')
plt.ylabel('Average Number of Bags')
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [1222]:
```
# Average Number of Bags by Country of Origin with Pie Chart

sizes = average_bags_per_origin['Number.of.Bags']
labels = average_bags_per_origin['Country.of.Origin']
explode = [0.1 if i == 0 else 0 for i in range(len(sizes))]
```

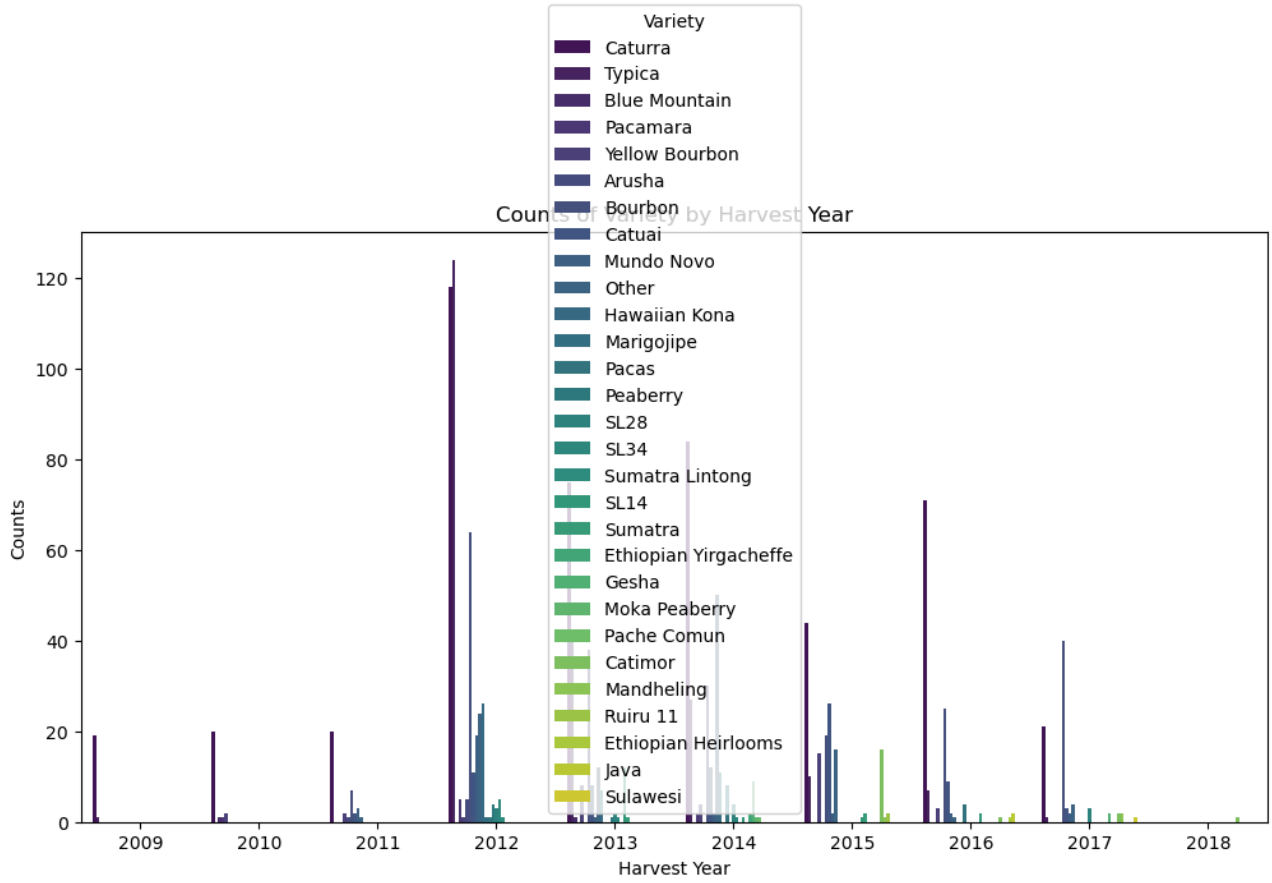
```
plt.figure(figsize=(10, 10))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, explode=explode)
plt.title('Average Number of Bags by Country of Origin')

plt.show()
```



Combination Viz 3 : *Let's Combine "**Harvest.Year**" and "**Variety**" for some visualization.*

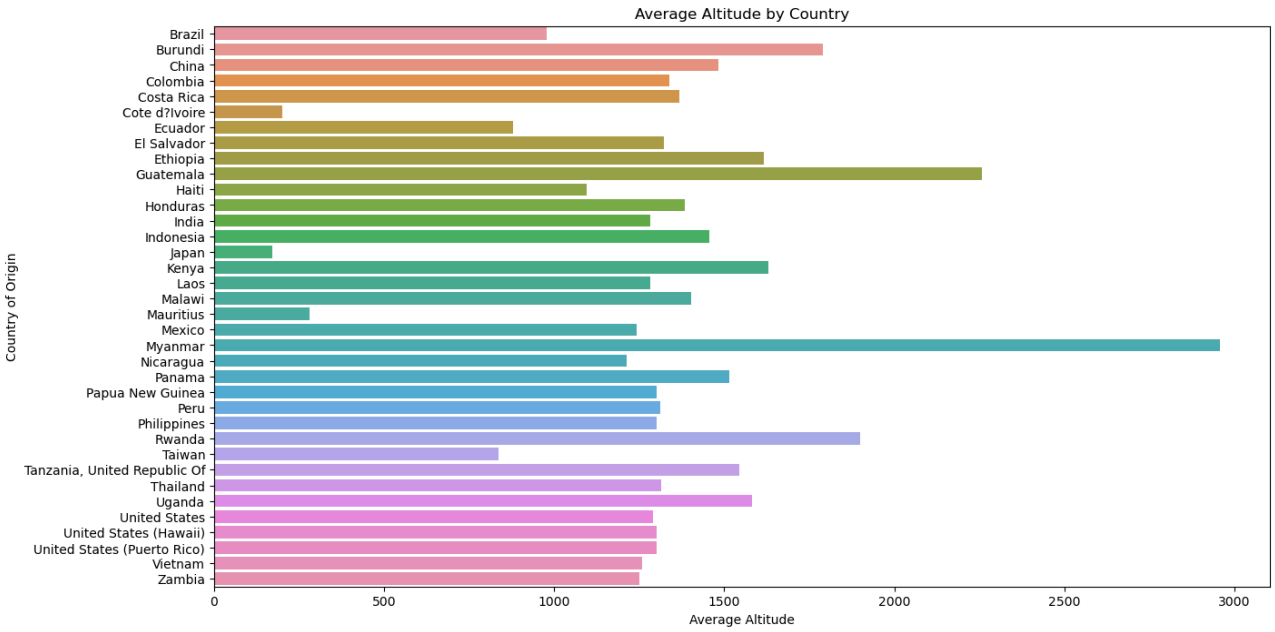
In [1251]:

```
# Count occurrences of each farm.name for each Harvest.Year
farm_year_counts = df4.groupby(['Harvest.Year', 'Variety']).size().reset_index(name='counts')

# Create a bar plot
plt.figure(figsize=(12,6))
sns.barplot(data=farm_year_counts, x='Harvest.Year', y='counts', hue='Variety', palette='viridis')

# Add labels and title
plt.xlabel('Harvest Year')
plt.ylabel('Counts')
plt.title('Counts of Variety by Harvest Year')
plt.show()
```

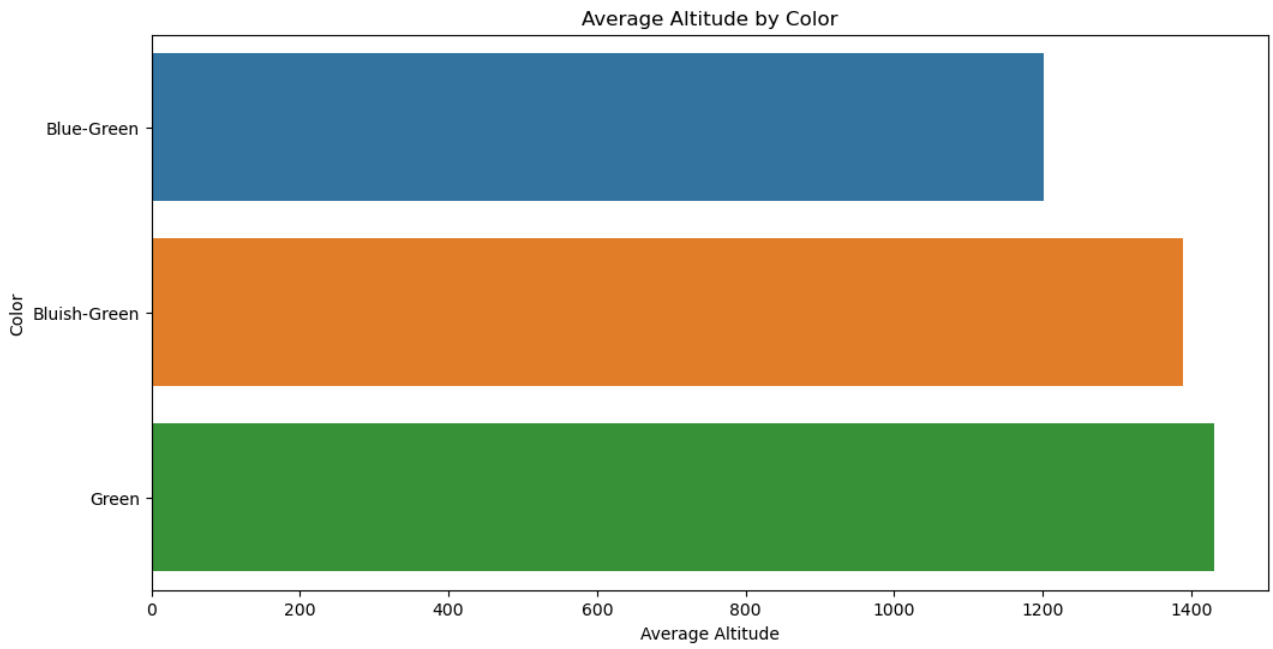


Combination Viz 4 : *Let's Combine columns with "**Altitude**" for some visualization and analysing*

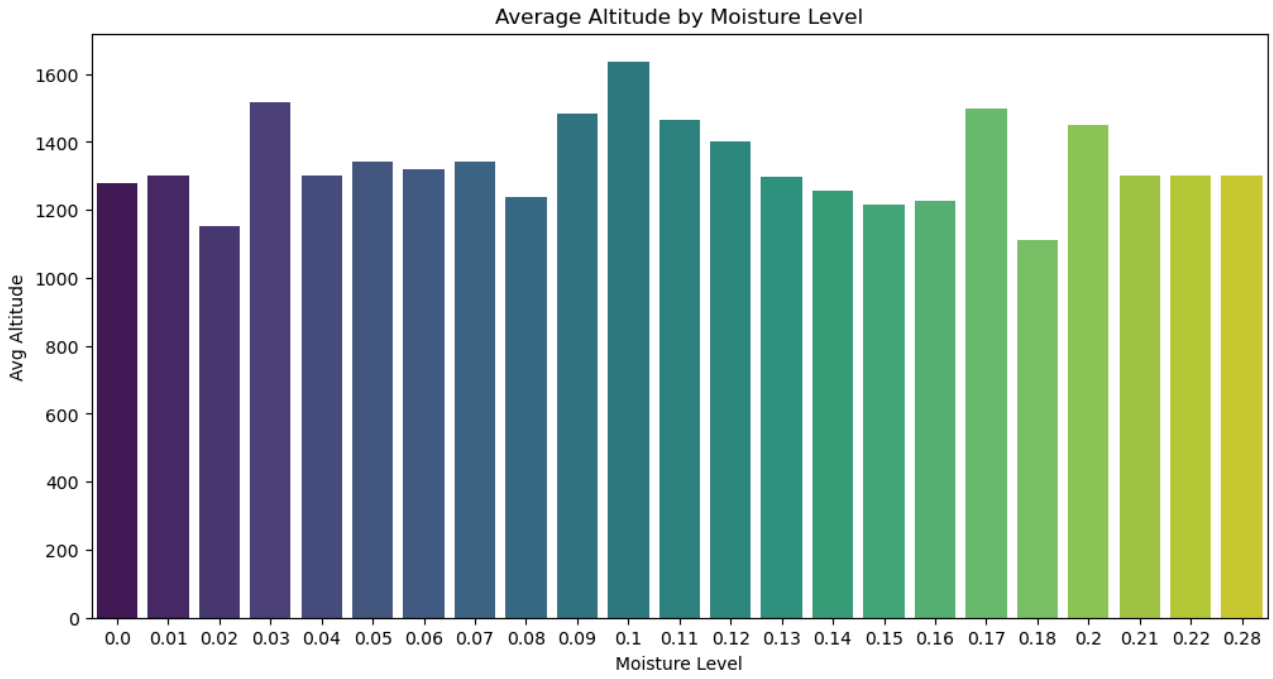
In [1270]:
```
# Average altitudes of coffee production in countries of origin
avg_altitude_by_country = df4.groupby('Country.of.Origin')['Altitude'].mean().reset_index()

plt.figure(figsize=(15, 8))
sns.barplot(x='Altitude', y='Country.of.Origin', data=avg_altitude_by_country)
plt.title('Average Altitude by Country')
plt.xlabel('Average Altitude')
plt.ylabel('Country of Origin')
plt.show()
```



In [1272]:
```
# Does altitude affect coffee color?
avg_altitude_by_color = df4.groupby('Color')['Altitude'].mean().reset_index()

plt.figure(figsize=(12, 6))
sns.barplot(x='Altitude', y='Color', data=avg_altitude_by_color)
plt.title('Average Altitude by Color')
plt.xlabel('Average Altitude')
plt.ylabel('Color')
plt.show()
```

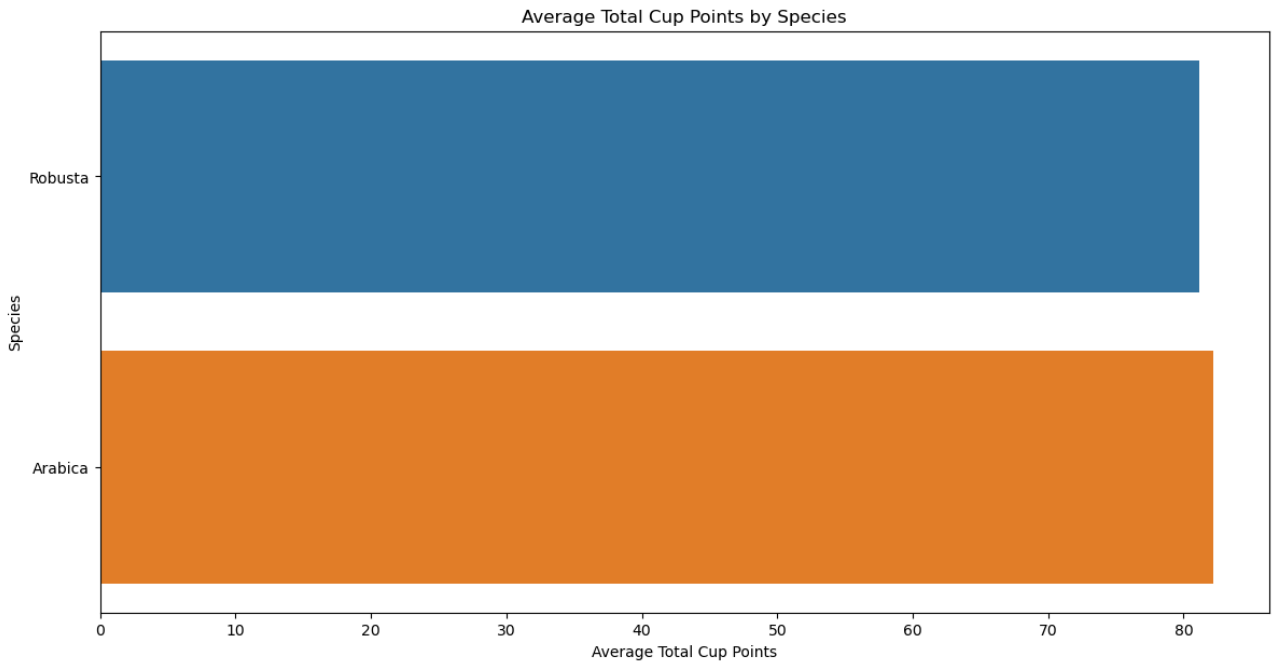


In [1282]:
```
# Calculate the average altitude by moisture level
avg_altitude_by_moisture = df4.groupby('Moisture')['Altitude'].mean().reset_index()

# Create the bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x='Moisture', y='Altitude', data=avg_altitude_by_moisture, palette='viridis')
plt.title('Average Altitude by Moisture Level')
plt.xlabel('Moisture Level')
plt.ylabel('Avg Altitude')
plt.show()
```

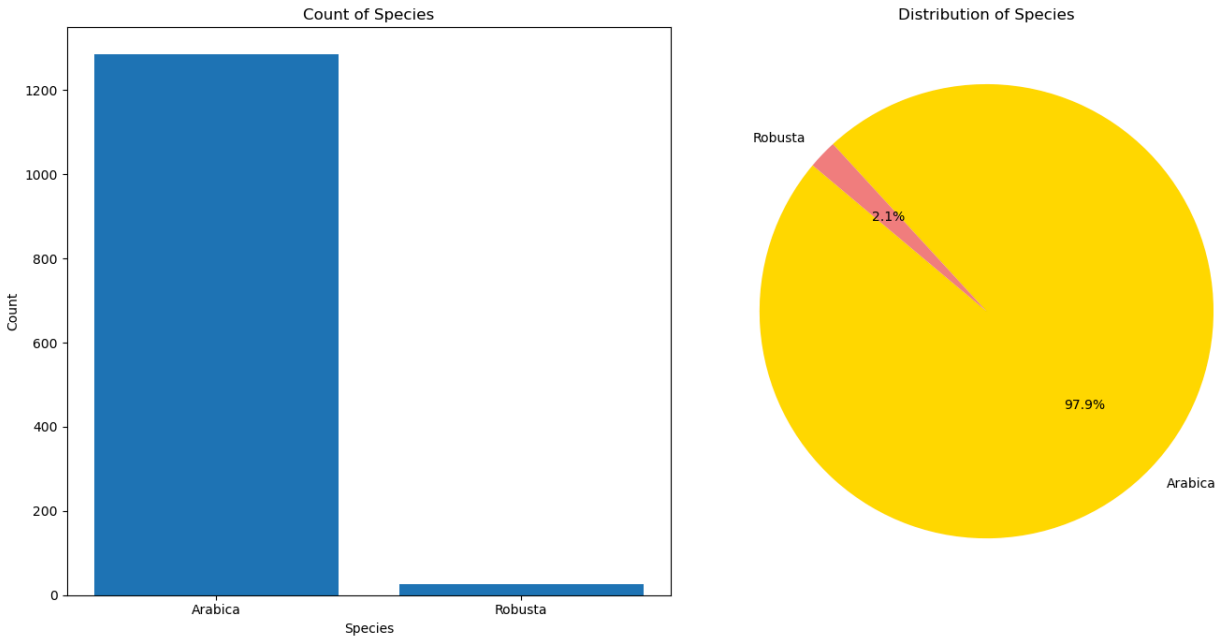


Combination Viz 5 : *Let's Combine **"Species"** and **"Total.Cup.Points"** for some visualization.*

In [1387]:
```
# Average Total Cup Points by Species
avg_total_cup_points_by_species = df4.groupby('Species')['Total.Cup.Points'].mean().reset_index()
plt.figure(figsize=(14,7))
sns.barplot(x='Total.Cup.Points', y='Species', data=avg_total_cup_points_by_species, order=avg_total_cup_points
plt.title('Average Total Cup Points by Species')
plt.xlabel('Average Total Cup Points')
plt.ylabel('Species')
Spect1=plt.gcf()
plt.show()
```

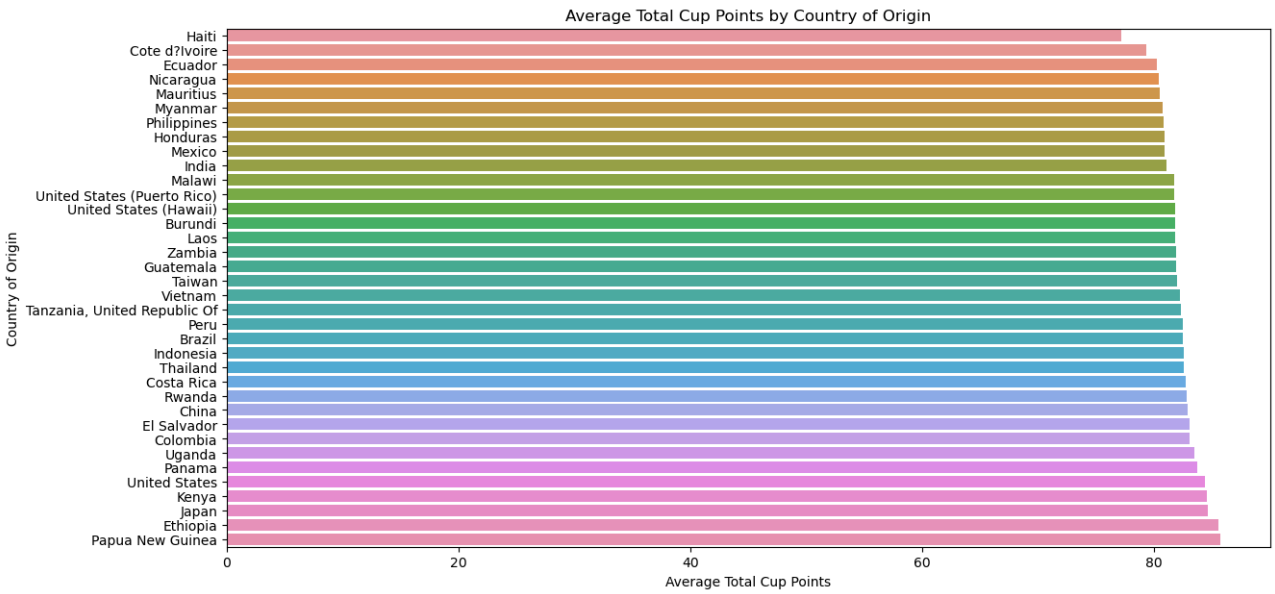


In [1381]:
```
# Create the figure and axes
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
# Bar chart (unchanged)
axs[0].bar(fre_tab['Species'], fre_tab['Count'])
axs[0].set_title('Count of Species')
axs[0].set_xlabel('Species')
axs[0].set_ylabel('Count')
# Define colors for the pie chart
pie_colors = ['gold', 'lightcoral', 'lightskyblue', 'lightgreen', 'pink']
# Pie chart with specified colors
axs[1].pie(fre_tab['Count%'], labels=fre_tab['Species'], autopct='%1.1f%%', startangle=140, colors=pie_colors)
axs[1].set_title('Distribution of Species')
# Ensure tight layout
plt.tight_layout()
# Save the figure to a variable named 'Spect'
Spect = plt.gcf()
plt.show()
```



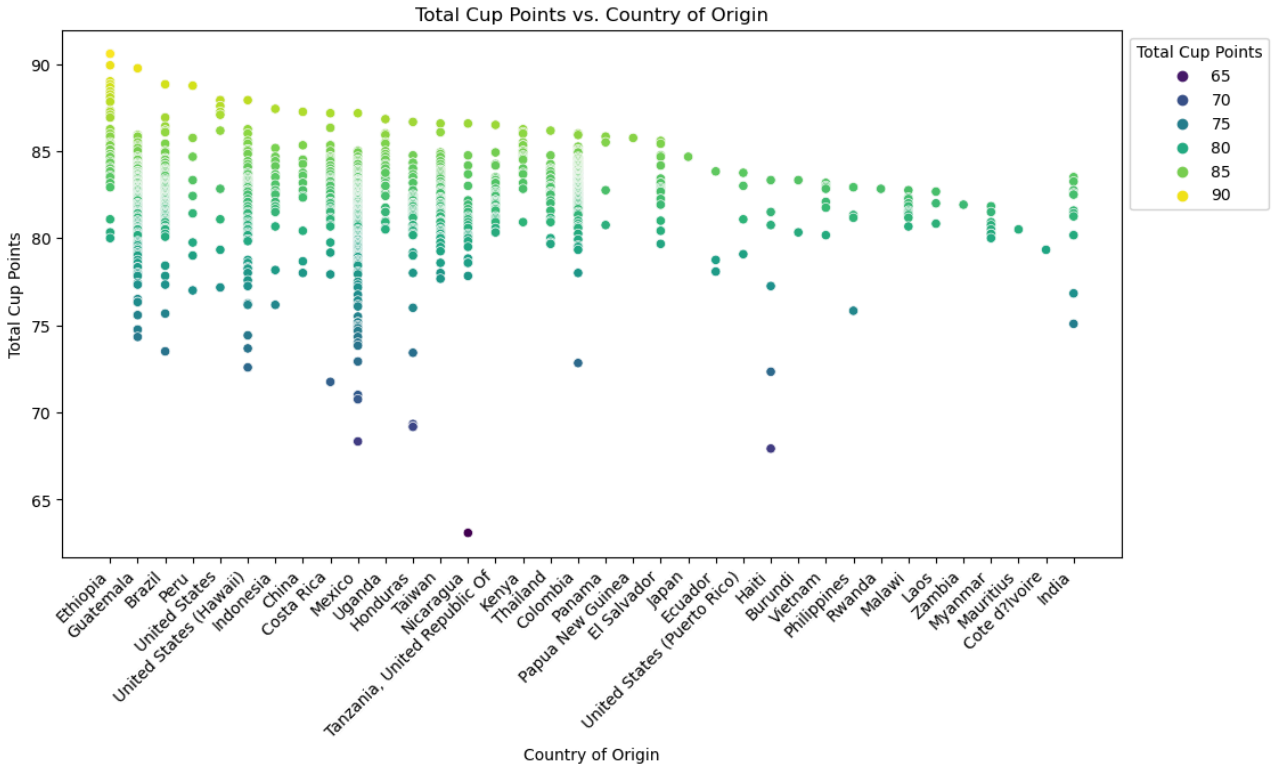
Combination Viz 6 : *Let's Combine "**Country.of.Origin**" and "**Total.Cup.Points**" for some visualization.*

In [1211]:
```
# Average Total Cup Points by Country of Origin
avg_total_cup_points_by_country = df4.groupby('Country.of.Origin')['Total.Cup.Points'].mean().reset_index()

plt.figure(figsize=(14,7))
sns.barplot(x='Total.Cup.Points', y='Country.of.Origin', data=avg_total_cup_points_by_country, order=avg_total_
plt.title('Average Total Cup Points by Country of Origin')
plt.xlabel('Average Total Cup Points')
plt.ylabel('Country of Origin')
plt.show()
```



In [1257]:

```
color_palette = sns.color_palette("viridis", as_cmap=True)
# Scatterplot
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Country.of.Origin', y='Total.Cup.Points', data=df4, hue='Total.Cup.Points', palette=color_pa
plt.title('Total Cup Points vs. Country of Origin')
plt.xlabel('Country of Origin')
plt.ylabel('Total Cup Points')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Total Cup Points', loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
```



In [ ]:

```
# Average Total Cup Points by Country of Origin
avg_total_cup_points_by_country = df.groupby('Country.of.Origin')['Total.Cup.Points'].mean().reset_index()

plt.figure(figsize=(14,7))
sns.barplot(x='Total.Cup.Points', y='Country.of.Origin', data=avg_total_cup_points_by_country, order=avg_total_
plt.title('Average Total Cup Points by Country of Origin')
plt.xlabel('Average Total Cup Points')
plt.ylabel('Country of Origin')
plt.show()
```

In [1049]:

```
import plotly.express as px
# Average Total Cup Points by Country of Origin
fig = px.choropleth(data,
                    locations='Country.of.Origin',
                    locationmode='country names',
                    color='Total.Cup.Points',
```

```
                    hover_name='Country.of.Origin',
                    color_continuous_scale=px.colors.sequential.Viridis,
                    title='Average Total Cup Points by Country of Origin')

fig.show()
```

IV) Analysis and Interpretation:

- Interpret findings from the data exploration and visualization, providing insights into factors influencing coffee quality.
- Identify key trends, patterns, or outliers that may impact quality assessment, production processes, or sourcing decisions.
- Offer recommendations or suggestions based on the analysis, such as potential areas for improvement in production methods, selection of bean varieties, or optimization of farming practices.

In [ ]:

V) Documentation:

- Document the entire data exploration and analysis process, including methodologies, findings, and interpretations.
- Present the analysis report in a clear and concise manner, catering to both technical and non-technical audiences.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

MARKDOWN VALUES USED IN THE CODE:

In [1128]:
```
from IPython.display import Markdown, display
bold_text = "**Summary Statistics of the Dataset :**"
bold_text1 = "**Missing Values :**"
bold_text2 = "**Percentage of Null-Values:**"
bold_text3 = "**Numerical Columns in the Dataset:**"
bold_text4 = "**Non-Numerical Columns in the Dataset:**"
bold_text5 = "**Total No. of Columns:**"
bold_text6 = "**Before Standardizing :**"
bold_text7 = "**After Standardizing :**"
bold_text8 = "**Outlier Detection in ScatterPlot**"
bold_text9 = "**Outliers in the Columns (Index)**"
bold_text10 = "**Data's Shape Before & After removal of Outliers**"
bold_text11 = "**Summary statistics of the new derived metrics :**"
bold_text12 = "**ANOVA FClass Features Results**"
```

Extra df for VISUALISATION

In [1417]:
```
df4 = data.copy()
```

In [1419]:
```
# Removing the Columns
df4.drop('Lot.Number', axis=1, inplace=True) # null values are more than 30% (i.e 79.39%)
df4.drop('Unnamed: 0', axis=1, inplace=True) # unncessary column
df4.drop('ICO.Number', axis=1, inplace=True) # unncessary column
df4.drop('Certification.Body', axis=1, inplace=True) # unncessary column
df4.drop('Certification.Address', axis=1, inplace=True) # unncessary column
df4.drop('Certification.Contact', axis=1, inplace=True) # unncessary column
df4.drop('unit_of_measurement', axis=1, inplace=True) # unncessary column
df4.drop('altitude_low_meters', axis=1, inplace=True) # unncessary column
df4.drop('altitude_high_meters', axis=1, inplace=True) # unncessary column
df4.drop('altitude_mean_meters', axis=1, inplace=True) # unncessary column
```

In [1421]:
```
# Function to convert lbs to kg and handle uncertain values
def convert_to_kg(weight):
    if pd.isna(weight):
        return None
    parts = weight.split()
    if len(parts) == 2:
        value, unit = parts
        value = float(value)
        if unit == 'lbs':
            value *= 0.453592
    elif len(parts) == 1:
        value = float(parts[0])
        # Assuming ambiguous values are in kg
        # Modify this according to your specific context
    else:
        value = None  # handle unexpected formats
    return value

# Apply conversion
df4['Bag.Weight'] = df4['Bag.Weight'].apply(convert_to_kg)
```

In [1423]:
```
import re
def extract_year(value):
    # Ensure the value is treated as a string
    if pd.isna(value):
        return None
    value = str(value)
    # Regular expression to find year patterns
    year_pattern = re.compile(r'(\b\d{4}\b)')
    match = year_pattern.search(value)
    if match:
        return match.group(1)
    # Handle cases like 'Mar-10' (assume it's 2010)
    if re.match(r'^[A-Za-z]{3}-\d{2}$', value):
        return '20' + value[-2:]
    # Handle cases like 'May-August' (assuming no specific year provided, default to NaN or any specific year
    return None

# Apply the function to the 'Harvest.Year' column
df4['Harvest.Year'] = df4['Harvest.Year'].apply(extract_year)
```

In [1425]:
```
# Since we already have - re library
# Function to clean and standardize the 'Altitude' column
def clean_and_standardize(altitude):
    if pd.isna(altitude):
        return np.nan  # Return NaN for missing values
    # Ensure the value is a string
    altitude = str(altitude)
    # Remove 'm', 'meters', and extra spaces
    altitude = re.sub(r'\s*m\s*|meters', '', altitude).strip()
    # Split by '-' and calculate the average if it's a range
    if '-' in altitude:
        try:
            parts = list(map(float, altitude.split('-')))
            return sum(parts) / len(parts)
        except ValueError:
            return np.nan
    # Convert single numeric values to float
```

```
    try:
        return float(altitude)
    except ValueError:
        return np.nan

# Apply the function to the 'Altitude' column
df4['Altitude'] = df4['Altitude'].apply(clean_and_standardize)
```

In [1427]:
```
# Replacement by median for Numerical Data USING Median and Non-Numerical Data USING Mode :
for column in All_Columns:
    if df4[column].dtype == 'object':
        mode_value = df4[column].mode()[0]
        print(f"Filling missing values in column '{column}' with mode: {mode_value}")
        df4[column].fillna(mode_value, inplace=True)
    else:
        median_value = df4[column].median()
        print(f"Filling missing values in column '{column}' with median: {median_value}")
        df4[column].fillna(median_value, inplace=True)
```

```
Filling missing values in column 'Species' with mode: Arabica
Filling missing values in column 'Owner' with mode: juan luis alvarado romero
Filling missing values in column 'Country.of.Origin' with mode: Mexico
Filling missing values in column 'Farm.Name' with mode: various
Filling missing values in column 'Mill' with mode: beneficio ixchel
Filling missing values in column 'Company' with mode: unex guatemala, s.a.
Filling missing values in column 'Altitude' with median: 1300.0
Filling missing values in column 'Region' with mode: huila
Filling missing values in column 'Producer' with mode: La Plata
Filling missing values in column 'Number.of.Bags' with median: 175.0
Filling missing values in column 'Bag.Weight' with median: 45.3592
Filling missing values in column 'In.Country.Partner' with mode: Specialty Coffee Association
Filling missing values in column 'Harvest.Year' with mode: 2012
Filling missing values in column 'Grading.Date' with mode: July 11th, 2012
Filling missing values in column 'Owner.1' with mode: Juan Luis Alvarado Romero
Filling missing values in column 'Variety' with mode: Caturra
Filling missing values in column 'Processing.Method' with mode: Washed / Wet
Filling missing values in column 'Aroma' with median: 7.58
Filling missing values in column 'Flavor' with median: 7.58
Filling missing values in column 'Aftertaste' with median: 7.42
Filling missing values in column 'Acidity' with median: 7.58
Filling missing values in column 'Body' with median: 7.5
Filling missing values in column 'Balance' with median: 7.5
Filling missing values in column 'Uniformity' with median: 10.0
Filling missing values in column 'Clean.Cup' with median: 10.0
Filling missing values in column 'Sweetness' with median: 10.0
Filling missing values in column 'Cupper.Points' with median: 7.5
Filling missing values in column 'Total.Cup.Points' with median: 82.5
Filling missing values in column 'Moisture' with median: 0.11
Filling missing values in column 'Category.One.Defects' with median: 0.0
Filling missing values in column 'Quakers' with median: 0.0
Filling missing values in column 'Color' with mode: Green
Filling missing values in column 'Category.Two.Defects' with median: 2.0
Filling missing values in column 'Expiration' with mode: December 26th, 2014
```

In [1429]:
```
# List of indices to drop
indices_of_outliers = [133, 179, 220, 734, 59, 88, 158, 224, 255, 420, 443, 843, 974, 1005, 1139, 896, 139, 234
# Drop the rows with the specified indices
df4 = df4.drop(index=indices_of_outliers)
```

In [ ]:

In [ ]:

In [ ]: