Coles Grocery Dataset

Objective

To analyze grocery retail data from Coles Australian supermarket, focusing on price distribution, product categories, and factors affecting retail prices.

Step 1: Importing Libraries and Loading Dataset

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np

# Load the dataset
df = pd.read_csv('Australia_Grocery_2022Sep.csv')

# Display first few rows of the dataset to understand its structure
df.head()
```

```
   index  Postal_code        Category Sub_category    Product_Group  \
0      0         2044  Meat & seafood      Poultry    Chicken offal
1      1         2044  Meat & seafood      Poultry    Chicken offal
2      2         2044  Meat & seafood      Poultry    Chicken offal
3      3         2044  Meat & seafood      Poultry    Chicken offal
4      4         2044  Meat & seafood      Poultry  Crumbed chicken

                          Product_Name  Package_price
Price_per_unit  \
0         RSPCA Approved Chicken Necks           7.15  $6.50 per 1Kg

1        RSPCA Approved Chicken Livers           5.40  $9.00 per 1Kg

2       RSPCA Approved Chicken Giblets           4.50  $7.50 per 1Kg

3        RSPCA Approved Chicken Frames           3.38  $4.50 per 1Kg

4  RSPCA Chicken Schnitzel Plain Crumb          10.50  $8.75 per 1Kg


        package_size  is_estimated  ...  Retail_price  \
0      approx. 1.1kg             1  ...           NaN
1       approx. 600g             1  ...           NaN
2       approx. 600g             1  ...           NaN
3  approx. 750g each             1  ...          4.13
```

```
4             1.2kg            0  ...        11.00

                                         Product_Url  Brand       Sku  \
0  https://shop.coles.com.au/a/alexandria/product...  Coles  1491280P   

1  https://shop.coles.com.au/a/alexandria/product...  Coles  1718058P   

2  https://shop.coles.com.au/a/alexandria/product...  Coles  2565429P   

3  https://shop.coles.com.au/a/alexandria/product...  Coles  3199541P   

4  https://shop.coles.com.au/a/alexandria/product...  Coles  2904193P   


              RunDate unit_price unit_price_unit  state   city       tid  
0  2022-11-09 08:23:06       6.50             1Kg    NSW  TEMPE  29742568  
1  2022-11-09 08:23:06       9.00             1Kg    NSW  TEMPE  29742569  
2  2022-11-09 08:23:06       7.50             1Kg    NSW  TEMPE  29742570  
3  2022-11-09 08:23:06       4.50             1Kg    NSW  TEMPE  29742571  
4  2022-11-09 08:23:06       8.75             1Kg    NSW  TEMPE  29742572  

[5 rows x 22 columns]
```

Step 2: Data Exploration - Missing Values and Data Types

In this step, we aim to identify missing values and understand the data types of each column. This is a crucial part of the Exploratory Data Analysis (EDA) process as it helps in determining how to handle missing data and ensures the dataset is ready for further analysis.

```
# Check for missing values and data types
print(df.info())

# Check the number of missing values in each column
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 488640 entries, 0 to 488639
Data columns (total 22 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   index            488640 non-null  int64 
 1   Postal_code      488640 non-null  int64 
 2   Category         488640 non-null  object
 3   Sub_category     488640 non-null  object
 4   Product_Group    488640 non-null  object
 5   Product_Name     488640 non-null  object
 6   Package_price    487313 non-null  float64
 7   Price_per_unit   485498 non-null  object
 8   package_size     487869 non-null  object
 9   is_estimated     488640 non-null  int64
 10  is_special       488640 non-null  int64
 11  in_stock         36454 non-null   object
 12  Retail_price     136953 non-null  float64
 13  Product_Url      488640 non-null  object
 14  Brand            488640 non-null  object
 15  Sku              488640 non-null  object
 16  RunDate          488640 non-null  object
 17  unit_price       485498 non-null  float64
 18  unit_price_unit  485498 non-null  object
 19  state            488640 non-null  object
 20  city             488640 non-null  object
 21  tid              488640 non-null  int64
dtypes: float64(3), int64(5), object(14)
memory usage: 82.0+ MB
None
index                  0
Postal_code            0
Category               0
Sub_category           0
Product_Group          0
Product_Name           0
Package_price       1327
Price_per_unit      3142
package_size         771
is_estimated           0
is_special             0
in_stock          452186
Retail_price      351687
Product_Url            0
Brand                  0
Sku                    0
RunDate                0
unit_price          3142
unit_price_unit     3142
state                  0
city                   0
tid                    0
dtype: int64
```

The dataset contains 488,640 rows and 22 columns. Below is a summary of the data types and the presence of non-null values for each column:

Numerical Columns: There are five numerical columns: index, Postal_code, Package_price, Retail_price, and unit_price. These columns are crucial for quantitative analysis.

Categorical Columns: The remaining columns, such as Category, Product_Group, Brand, and city, are represented as object types, which will be useful for segmentation and categorical analysis.

Retail_price: 351,687 missing values.

in_stock: 452,186 missing values, which suggests that only a small fraction of the data includes stock information.

Price_per_unit and unit_price: Both have around 3,142 missing values, which can affect price-related analyses.

package_size: 771 missing values.

Step 3: Data Cleaning and Imputation

In this section, we address missing values and standardize package size units to ensure consistency across the dataset. This process is essential for ensuring that our dataset is ready for analysis.

Imputing Missing Values: To handle missing values in continuous variables, we use the median value. The median is a robust measure of central tendency, especially when dealing with skewed distributions, which is the case for retail prices in this dataset.

```
# Fill missing Retail_price with median as it's a continuous variable
df['Retail_price'].fillna(df['Retail_price'].median(), inplace=True)

# Fill missing unit_price with the median value as well
df['unit_price'].fillna(df['unit_price'].median(), inplace=True)
```

Retail Price: The missing values in the Retail_price column are filled with the median retail price of the dataset. This ensures that we retain as many data points as possible for analysis without introducing outliers.

Unit Price: Similarly, missing values in the unit_price column are replaced by the median of the column.

Standardizing Package Sizes: The package_size column contains a mix of units (grams and kilograms) and non-numeric information such as "approx." or "each." We create a function to clean and standardize package sizes by converting everything to grams for consistency.

```
# Improved function to convert package sizes to grams
def convert_package_size(size):
    try:
        # Remove any non-numeric words like 'approx.', 'each', etc.
        size = size.lower().replace('approx.', '').replace('each',
'').strip()
```

```
        if 'kg' in size:
            return float(size.replace('kg', '').strip()) * 1000  # 
Convert Kg to grams
        elif 'g' in size:
            return float(size.replace('g', '').strip())  # Leave grams 
as is
        else:
            return None  # Handle unknown formats
    except:
        return None

# Apply the function to clean package sizes
df['package_size_standard'] = df['package_size'].apply(lambda x: 
convert_package_size(str(x)))

# Check the cleaned data again
df[['package_size', 'package_size_standard']].head()
```

```
        package_size  package_size_standard
0      approx. 1.1kg                 1100.0
1       approx. 600g                  600.0
2       approx. 600g                  600.0
3  approx. 750g each                  750.0
4              1.2kg                 1200.0
```

Original Package Size: The original package_size column contains text with different units.

Standardized Package Size: The new package_size_standard column holds cleaned and standardized values, converting everything into grams. This will allow for easy comparison of package sizes across products.

Key Insights After Data Cleaning:

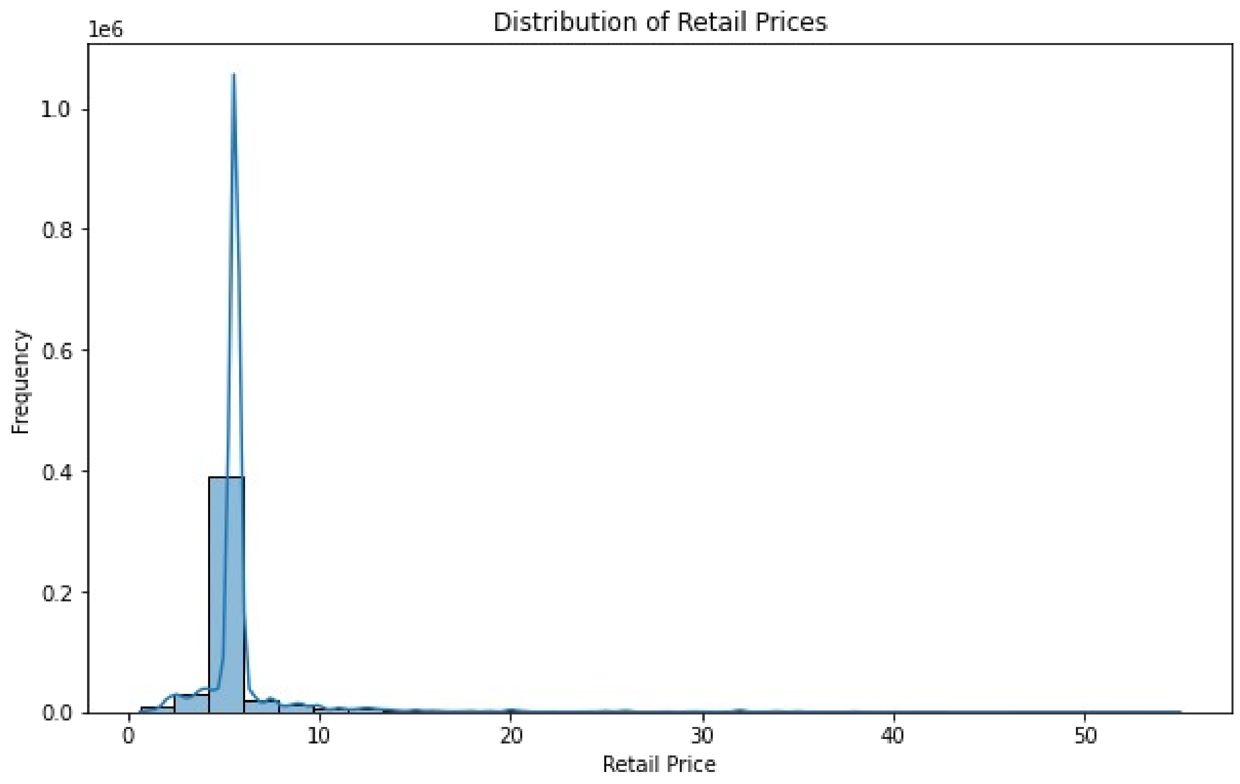
Consistency Across Package Sizes: By converting all package sizes to grams, we ensure that comparisons between different products are consistent and meaningful.

Handling Missing Values: Filling missing Retail_price and unit_price values with the median ensures that our dataset is as complete as possible without distorting the analysis with outliers or extreme values.

Step 4: Exploratory Data Analysis (EDA) - Distribution of Retail Prices

In this plot, we are examining the distribution of retail prices in the dataset. A histogram is created to understand how frequently certain price ranges occur, and a Kernel Density Estimate (KDE) is applied to visualize the distribution smoothly.

```
# # 1. Distribution of Retail Prices
plt.figure(figsize=(10,6))
sns.histplot(df['Retail_price'], kde=True, bins=30)
plt.title('Distribution of Retail Prices')
plt.xlabel('Retail Price')
plt.ylabel('Frequency')
plt.show()
```



Key Insights:

Right-Skewed Distribution: Most of the retail prices are clustered in the lower price range (around 0 to 10 units), with a steep drop-off for higher prices.

Majority of Products: A very large number of products have retail prices concentrated between 0 and 5 units of currency.

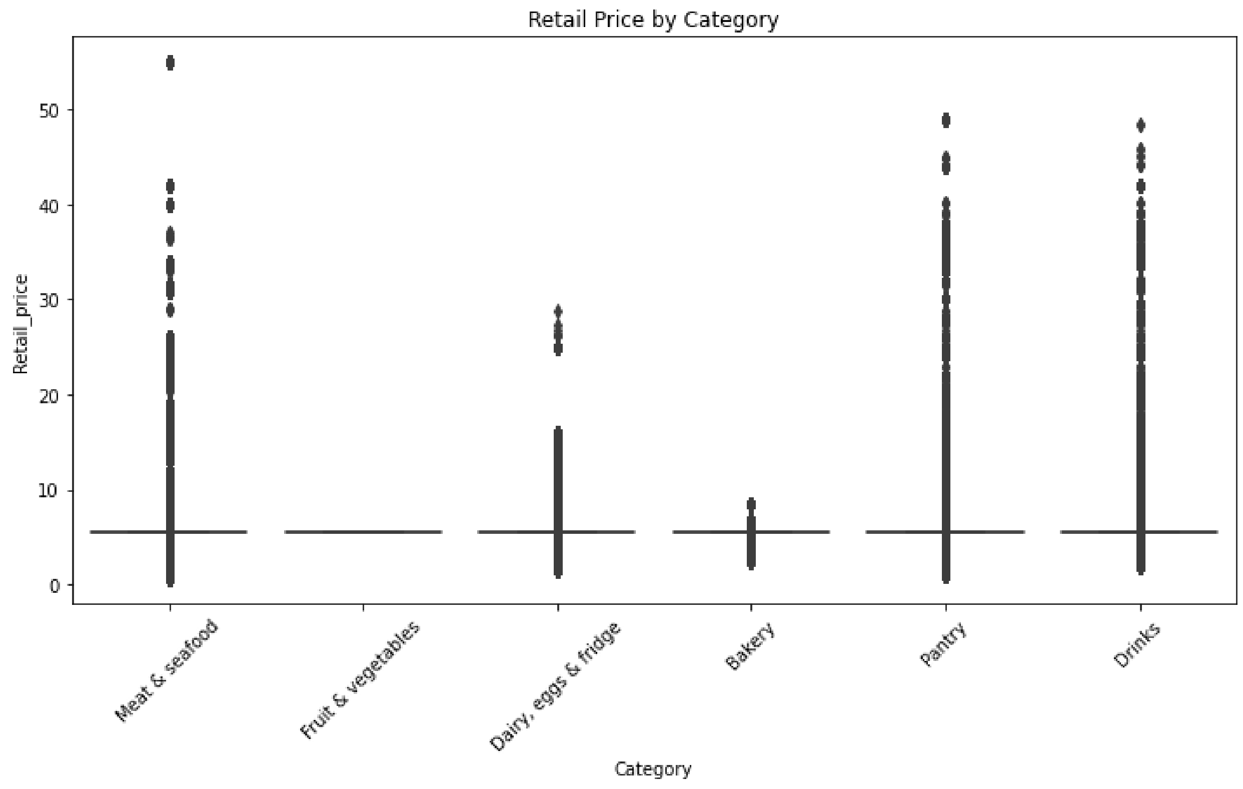
Few High-Priced Products: As the prices increase beyond 10, the number of products decreases significantly, and there are very few products priced beyond 50 units.

This plot suggests that most products in the dataset are relatively affordable, with only a small fraction of premium-priced items. This kind of distribution is common in grocery datasets where the majority of goods are priced for everyday purchases.

Boxplot of Retail Prices by Category

In this boxplot, we are comparing the distribution of retail prices across different product categories. Boxplots are useful for visualizing the spread of the data and identifying outliers.

```
# 2. Boxplot of Retail Prices by Category
plt.figure(figsize=(12,6))
sns.boxplot(x='Category', y='Retail_price', data=df)
plt.xticks(rotation=45)
plt.title('Retail Price by Category')
plt.show()
```



Key Insights:

Meat & Seafood:

- This category has a wide price range, with many outliers that extend above 50 units of currency, indicating premium or high-priced products.
- The median price seems moderate, but the large spread suggests that there is a wide variation in pricing within this category.

Dairy, Eggs & Fridge: There is some price variability, but it is narrower than the Meat & Seafood category. A few outliers are present, with prices above 30 units.

Other Categories (e.g., Bakery, Pantry, Drinks):

- These categories show fewer extreme outliers, and their prices appear to be lower on average compared to the Meat & Seafood category.
- The price ranges for these categories are more compact, indicating more consistent pricing across products.

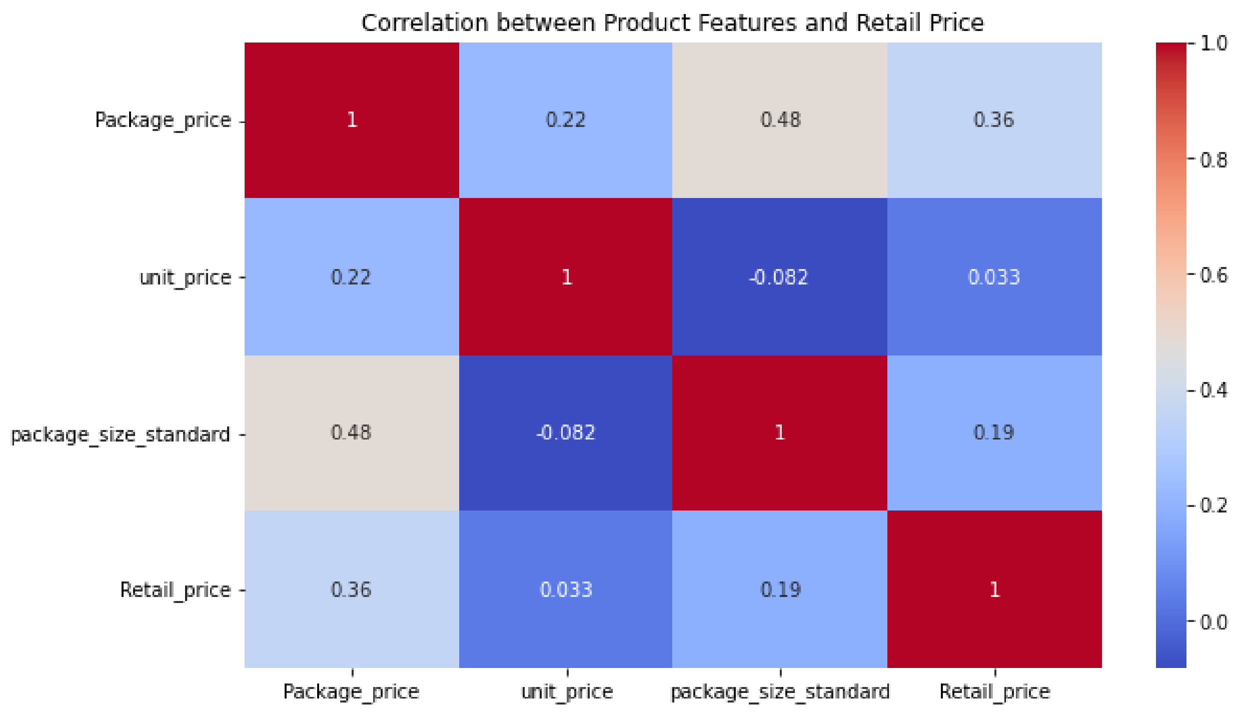
Outliers: The plot highlights several outliers (dots outside the whiskers), especially in the Meat & Seafood and Drinks categories, where some items are priced much higher than the rest.

Low Variability in Some Categories: Categories like Fruit & Vegetables and Bakery show very little price variation, suggesting that most of the products in these categories are similarly priced.

Correlation between Product Features and Retail Price

This heatmap visualizes the correlation between various numerical features in the dataset, including package price, unit price, package size (standardized to grams), and retail price. The aim is to understand the relationships between these variables and how they influence the retail price.

```python
# 3. Correlation Heatmap between numerical features
plt.figure(figsize=(10,6))
sns.heatmap(df[['Package_price', 'unit_price',
'package_size_standard', 'Retail_price']].corr(), annot=True,
cmap="coolwarm")
plt.title('Correlation between Product Features and Retail Price')
plt.show()
```



	Package_price	unit_price	package_size_standard	Retail_price
Package_price	1	0.22	0.48	0.36
unit_price	0.22	1	-0.082	0.033
package_size_standard	0.48	-0.082	1	0.19
Retail_price	0.36	0.033	0.19	1

The heatmap displays Pearson correlation coefficients between the features, with values ranging from -1 to 1:

1: Perfect positive correlation.

0: No correlation.

-1: Perfect negative correlation.

Key Insights:

Package Price vs. Retail Price:

Correlation: 0.36 – This shows a moderate positive correlation. As package price increases, retail price tends to increase as well, but the relationship is not very strong.

Package Size vs. Retail Price:

Correlation: 0.19 – A weak positive correlation exists between package size and retail price. Larger packages do tend to have higher prices, but this relationship is not consistent across all products.

Package Size vs. Package Price:

Correlation: 0.48 – This indicates a moderate positive correlation, suggesting that larger package sizes are generally more expensive.

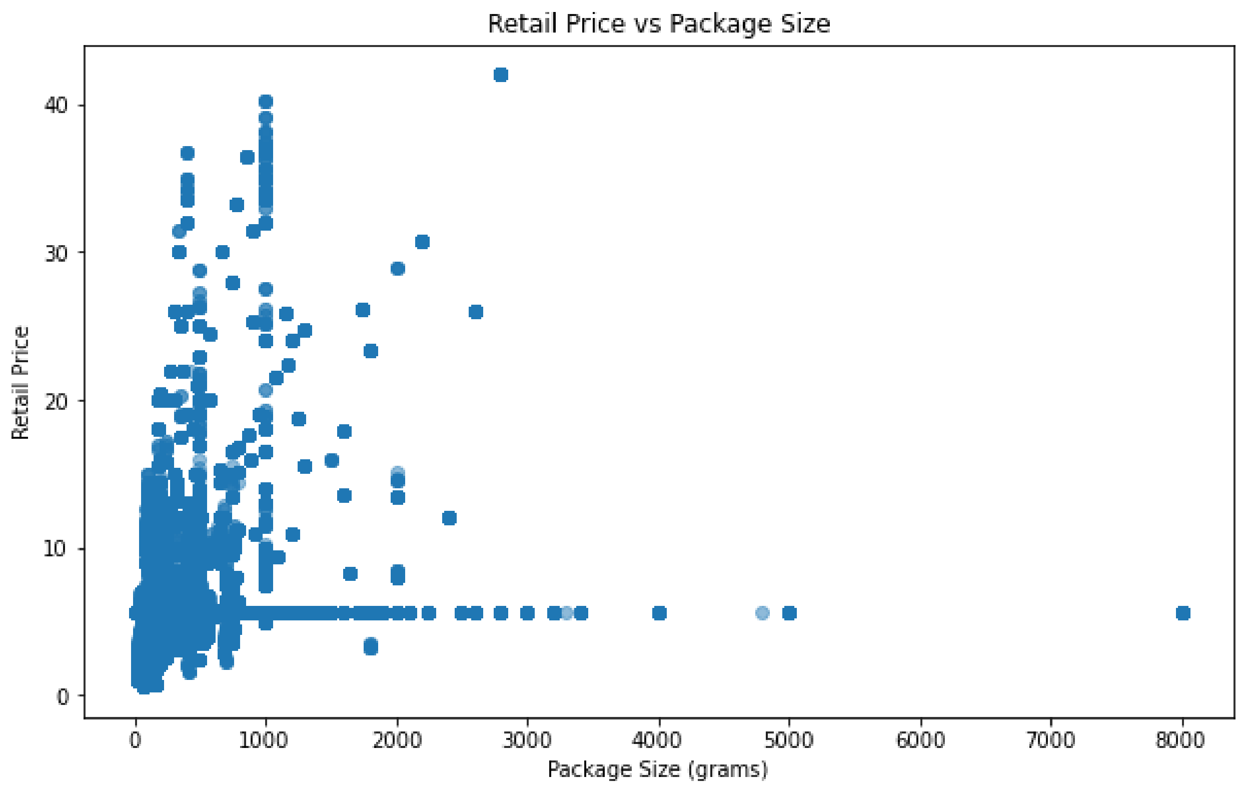
Unit Price: The correlations between unit price and the other features are weak, especially with retail price (0.033), which implies that unit price alone does not have a strong influence on retail price in this dataset.

Conclusion: The heatmap reveals that while package price and package size have moderate correlations with retail price, they are not the sole determinants of pricing. Unit price shows very little correlation with retail price, indicating that other factors (such as product category or promotional status) may play a larger role in price determination.

Scatter Plot: Retail Price vs Package Size

This scatter plot examines the relationship between package size (in grams) and retail price. It allows for the visualization of how product size affects pricing within the dataset.

```
# 4. Scatter plot of Retail Price vs Package Size
plt.figure(figsize=(10,6))
plt.scatter(df['package_size_standard'], df['Retail_price'],
alpha=0.5)
plt.title('Retail Price vs Package Size')
plt.xlabel('Package Size (grams)')
plt.ylabel('Retail Price')
plt.show()
```



Key Insights:

Cluster of Smaller Packages: The majority of products have small package sizes (below 1000 grams), and their prices vary widely. Most of the prices for these smaller packages are between 0 and 10 units.

Few Large Packages: There are very few products with package sizes above 2000 grams. The prices for these larger packages are relatively low, typically below 20 units, with a few exceptions.

Outliers: A few products with large package sizes (4000 grams and above) have significantly low retail prices, possibly indicating bulk or multi-pack products. There is also an outlier at around 40 units for a package size of approximately 1000 grams, potentially representing a premium product.

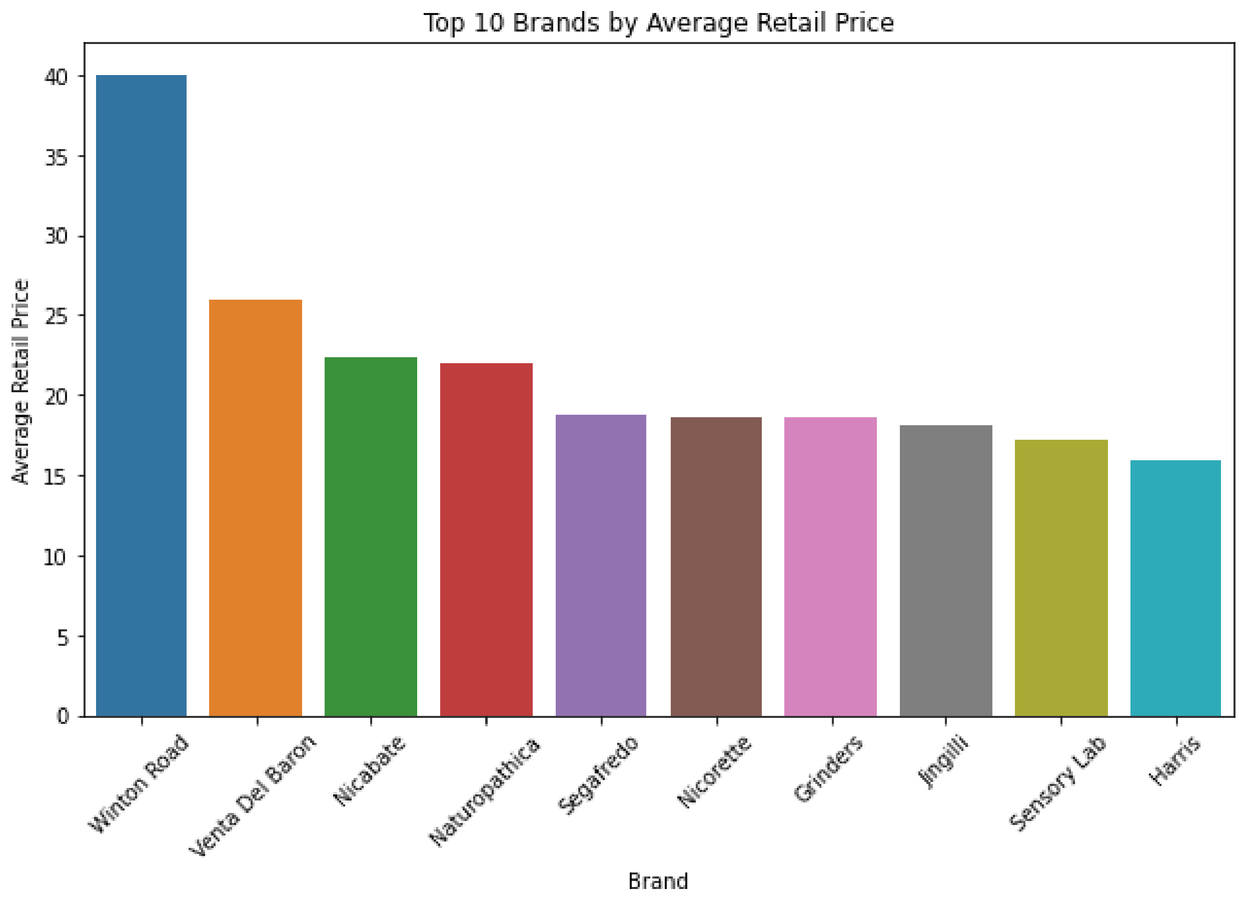
Weak Correlation: The scatter plot shows no clear linear relationship between package size and retail price, supporting the earlier correlation heatmap which indicated a weak correlation (0.19) between these two variables.

Conclusion: This plot suggests that package size alone does not strongly determine retail price. Many small packages are priced across a wide range, and larger packages do not necessarily command significantly higher prices. There may be other factors, such as product category or brand, that influence pricing decisions.

Bar Plot: Top 10 Brands by Average Retail Price

This bar plot visualizes the top 10 brands based on their average retail price. It ranks brands from the highest to the lowest average price.

```python
# 5. Average Retail Price by Brand (Top 10)
top_brands = df.groupby('Brand')
['Retail_price'].mean().sort_values(ascending=False).head(10)
plt.figure(figsize=(10,6))
sns.barplot(x=top_brands.index, y=top_brands.values)
plt.title('Top 10 Brands by Average Retail Price')
plt.xticks(rotation=45)
plt.ylabel('Average Retail Price')
plt.show()
```



Key Insights:

Winton Road: This brand has the highest average retail price, exceeding 40 units, suggesting it offers premium products compared to others.

Venta Del Baron: Following closely, this brand's average retail price is slightly above 30 units, indicating it too offers high-end or specialized products.

Mid-range Brands: Brands like Nicabate, Naturopathica, Segafredo, and Nicorette have average prices in the range of 20 to 25 units. These brands may offer a mix of mid-tier to premium products.

Lower-Priced Brands: Grinders, Jingilli, Sensory Lab, and Harris round out the top 10 with average prices just below 20 units. These brands, though in the top 10, offer more moderately priced products compared to Winton Road and Venta Del Baron.

Conclusion: This plot provides a clear comparison of premium and mid-range brands by average retail price. Brands like Winton Road and Venta Del Baron stand out as offering high-cost products, while others like Sensory Lab and Harris are more moderately priced.