Technical Report: Team 1

Analyst: **Luis Lopez and Mohammad Alkhawaldeh**

Date: 02/26/2024

Introduction:

This report presents an analysis of the wholesale data for a range of products. The objective is to understand the expenditure patterns of clients across different regions and product categories and to assist the stakeholders in understanding different metrics of the products being consumed. Furthermore, we elaborate on some startegies tom improve product sales, confidence intervals of the mean of the products being consumed and marketing suggestions.

Data Overview/Preparation:

The dataset contains information on the expenditure of clients on various products, including Fresh, Milk, Grocery, Frozen, Detergents_Paper, and Delicassen. The data is categorized by region and all of its content required no cleaning. It did not contain any Null values and and the data types were with their corresponding data value. The shape of the dataset had 400 rows and 8 columns (Appendix, 1-1), out of the 8 columns there are two nominal categorical variables and 6 quantitative continuous variables.

Analysis 1-1 - 1-8:

1-3, 1-4:

Region 3 has the highest total sales across all product categories at 73% of total sales. Focus marketing efforts here for its sales potential.. Region 1 has the second highest total sales at 16% of total sales. It should also receive marketing attention. Region 2 has the smallest share of total sales at 10%. It is a lower priority for marketing. Fresh and Grocery lead sales in all regions. Prioritize marketing campaigns toward these categories.

1-6

On average, customers spend the most on Fresh Food and Groceries. They spend a good amount on Milk and Frozen Food too. Marketing should focus more on these categories since customers buy them more. Customers buy very different amounts of each product. Some buy a lot, some buy only a little bit. Marketing should have offers for all customer groups - those who buy little, medium amounts, and large amounts. A small number of customers spend a huge amount in some categories. These customers are very valuable. The store should have special VIP rewards to keep these big spenders happy.

1-8:

The analysis shows a strong connection between spending on groceries and spending on detergents and paper products. Customers who put more in their grocery carts also tend to buy more laundry detergent, paper towels, and other cleaning supplies. This makes intuitive sense - the more you cook and prepare food at home, the more you need supplies to keep your home clean.

We also see that milk purchases are closely linked to grocery and detergent/paper spending. When customers buy more milk, they also spend more on other grocery staples and home cleaning products. Milk is a common purchase for most households, so higher milk spending likely indicates overall higher grocery spending.

Recommendations: Offer cross-promotions linking grocery and home cleaning staples to reflect how customers buy these together. For example, "Spend 50 M.U on Grocery and Get 5 M.U Off Detergent" Position milk where shoppers add a lot to their overall grocery carts.

Analysis 1-9 - 1-15

1-9 - 1-10:

These two lines of code represent a subset of our data, in order to utilize the the columns in a more efficient way for the next blocks of codes we decided to drop channel and region columns.

1-11 - 1-12:

We created a function to plot and calculate the confidence intervals for the categories of products we wanted to analyze. On block 1-7 we saw the distributions of each of the products and resembled a gamma distribution, therefore in order to construct a sampling distribution of these categories we decided to use the Gamma function with their corresponding parameters to plot the sampling distribution a long with the confidence intervals to have an idea of where the mean of each of the products being sold was going to land. The block also plots the approximation of the probability density function of the product category to confirm that we are representing the original population distribution. The libraries we used were Seaborn, matplot.lib, Numpy, and Stats.Gamma. Customers' behaviors and product spending follow predictable patterns that can be modeled mathematically. This allows reliable sales predictions. The models show most customers spend around typical averages, but some spend much less, and some spend way more.

1-13:

To check for independence of the two categorical variables we were working with we needed to create a contingency table to perform the Chi-squared test for independecnce. The code above shows how to create a contingency table. Fitted values in the context of a contingency table are the expected frequencies of observations for each cell of the table under the null hypothesis of independence between the variables. In other words, it shows what the distribution of 'Channel' across 'Region' would look like if there were no association between the two.

1-14:

This block of code we used the seaborn library to plot a correlation matrix to get a better understanding of the relationship of our features and to explore how strong their relations might be before we start building a regression model for prediction.Customers show different patterns of spending across categories. Most of them have moderate spending amounts, some spending very little, and a few spending large amount.Spending on certain product categories is proportional, with customers who spend more on groceries also spending more on related items like cleaning items.The business strategy should focus on regular shoppers and rewards programs for high-spenders, along with bundled promotions for related products like groceries and detergents.

1-15:

This is our first attempt to create a predicting equation for our target variable groceries. We chose groceries because the amount of sales that the Fresh product brought, most likely the trend was going to continue, but we if we bring in more of the other correlated products to predict the second highest bought product then we can strategize in maximize the related products. This Ordinary Least Square performed quite well using Detergents_paper and Milk as the predictors but then we remove Milk to find out that the R-squared value did not change significantly when we removed Milk. The block of code also uses another test, the AIC measured our models performance. The residual and normal Q-Q plots verify that the linear model has properly fitted the grocery spending data with no major abnormal assumptions. The errors and residuals indicate the model is trustworthy for insights and predictions.

1-16:

In comparison to the previous model, this time we used a Generalized Linear Model with the Gamma family and the identity link function to see how it performed in comparison to the OLS. The result was in favor of the GLM and the same formula was used as the OLS when we compared the AIC of both models. Another detail we noticed was that by adding milk in the GLM the AIC dropped a bit more so we decided to include it in our regression model as well

1-17:

Once we chose our regression model, we wanted to check how good our model was at predicting the values so we used seaborn and the ”predict” function to create a scatter plot against the actual Grocery values. We also added a “best fit line” which represented the perfect fit of the Grocery sales.The prediction dots and actual spending line are matching closely to the model and predicts pretty accurately for shoppers who spend around average.

1-18:

This last block of code is another scatter plot to see how the residuals of our model that we chose was performing against the fitted values or predicted values. The scatter plot shows variability when the fitted plots are converted to a log format. Generally we want to see that the points in the scatter plot to be spread and without a necessary pattern. In this care we see som dispersion among the points and seem to be clustering near the center of the x-axis or at 0. This suggest that our model still shows weakness in predicting values due to the nature of the distribution of the Groceries feature. A solution could have been to consider removing outliers from our data but, since we are working with sales, it is important to consider all points because there a vendors and regions that might do consume that amount of products. The residuals plot verifies the model fits well for average customer spending amounts in the majority middle range.

Conclusion:

In conclusion, our biggest obstacle was to work with a dataset that was highly skewed but at the end we were successful in visualizing the data and understanding how sales behave in a given region and channel. The skewness of the products allowed us to explore and apply the gamma function and taught us that even with highly skewed data we are able to extract useful information for business purposes. In our linear models, we were able to predict future sales of groceries using 2 other products which can help understand future sales and trends a given region or channels is experiencing. They can also now understand what are their confidence intervals and understand were their mean sales might fall on the long-run with 95% confidence. Finding out that channel and region were independent categorical variables will help marketing create a one-size fits all concept to all region and channels when creating a marketing campaign. Additionally we better understood the relationship between spending habits for each product consumed which allowed us to build a predictive model for Groceries, using the correlation of Detergents_Paper. This will allow the stakeholders have more control over a high influc of customers inthe future but it will also help them understand that there a customer that have a much higher need for the products.

Appendix

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
c:\Users\luisl\anaconda3\envs\PythonData\lib\site-packages\scipy\__init__.py:146: U
serWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of Sc
iPy (detected version 1.25.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Looks like the top selling products are Fresh, Milk and Groceries. Numbers are expenditure of clients in monetary units

Descriptive Statistics

1-1

In [ ]:
```
# Read CSV file into DataFrame
df = pd.read_csv('wholesale.csv')

# Display first 5 rows
print(df.head(5), "\n")

# Calculate percentage of each column sum
print((df.sum() / df.sum().sum()) * 100)

#shape of the Dataset
print(df.shape)
```

```
   Channel  Region  Fresh  Milk  Grocery  Frozen  Detergents_Paper  Delicassen
0        2       3  12669  9656     7561     214              2674        1338
1        2       3   7057  9810     9568    1762              3293        1776
2        2       3   6353  8808     7684    2405              3516        7844
3        1       3  13265  1196     4221    6404               507        1788
4        2       3  22615  5410     7198    3915              1777        5185

Channel              0.003981
Region               0.007653
Fresh               36.112841
Milk                17.442869
Grocery             23.928007
Frozen               9.244453
Detergents_Paper     8.671360
Delicassen           4.588836
dtype: float64
(440, 8)
```

1-2

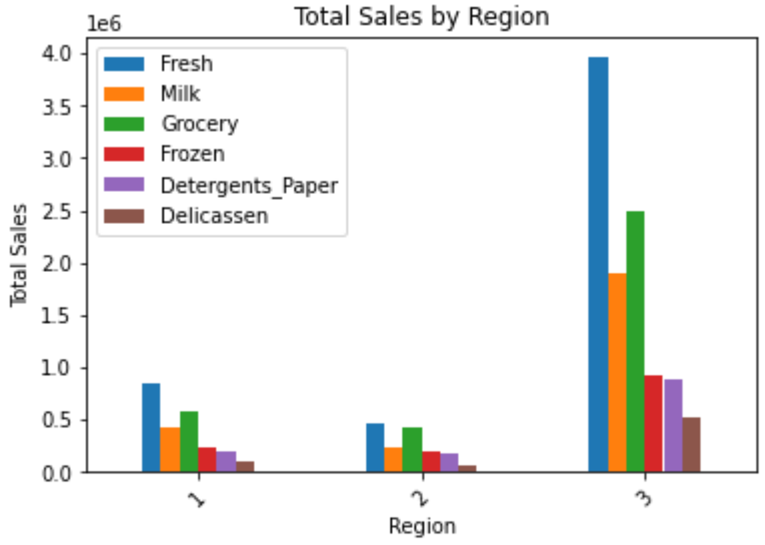
In [ ]:
```
df.info() # Display DataFrame information
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Channel           440 non-null    int64
 1   Region            440 non-null    int64
 2   Fresh             440 non-null    int64
 3   Milk              440 non-null    int64
 4   Grocery           440 non-null    int64
 5   Frozen            440 non-null    int64
 6   Detergents_Paper  440 non-null    int64
 7   Delicassen        440 non-null    int64
dtypes: int64(8)
memory usage: 27.6 KB
```

1-3

In [ ]:
```python
# Plot total sales of different product categories by region
plt.figure(figsize=(6,7))
df.groupby('Region')[['Fresh', 'Milk', 'Grocery', 'Frozen',
       'Detergents_Paper', 'Delicassen']].sum().plot(kind='bar')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)
plt.title('Total Sales by Region')

# Set labels for x and y axes
plt.xlabel('Region')
plt.ylabel('Total Sales')

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

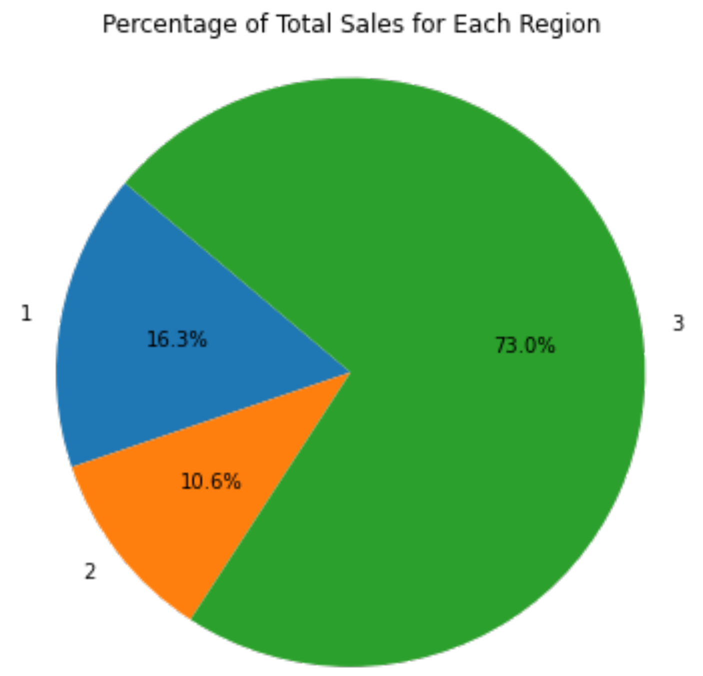
```
<Figure size 432x504 with 0 Axes>
```



1-4

In [ ]:
```python
# Calculate percentage of total sales for each region across all product categories
print(round(df.groupby('Region')[['Fresh', 'Milk', 'Grocery', 'Frozen',
       'Detergents_Paper', 'Delicassen']].sum().sum(axis=1) / df.sum().sum() * 100,

# Pie chart for percentage of total sales for each region
plt.figure(figsize=(8, 6))
region_sales_percentage = df.groupby('Region')[['Fresh', 'Milk', 'Grocery', 'Frozen
plt.pie(region_sales_percentage, labels=region_sales_percentage.index, autopct='%1.
plt.title('Percentage of Total Sales for Each Region')
plt.axis('equal')
plt.show()
```

```
Region
1    16.32
2    10.64
3    73.03
dtype: float64
```



1-5

In [ ]:

```python
# Calculate percentage of total sales for each channel across all product categorie
df.groupby('Channel')[['Fresh', 'Milk', 'Grocery', 'Frozen',
       'Detergents_Paper', 'Delicassen']].sum().sum(axis=1) / df.sum().sum() * 100
```

Out[ ]:

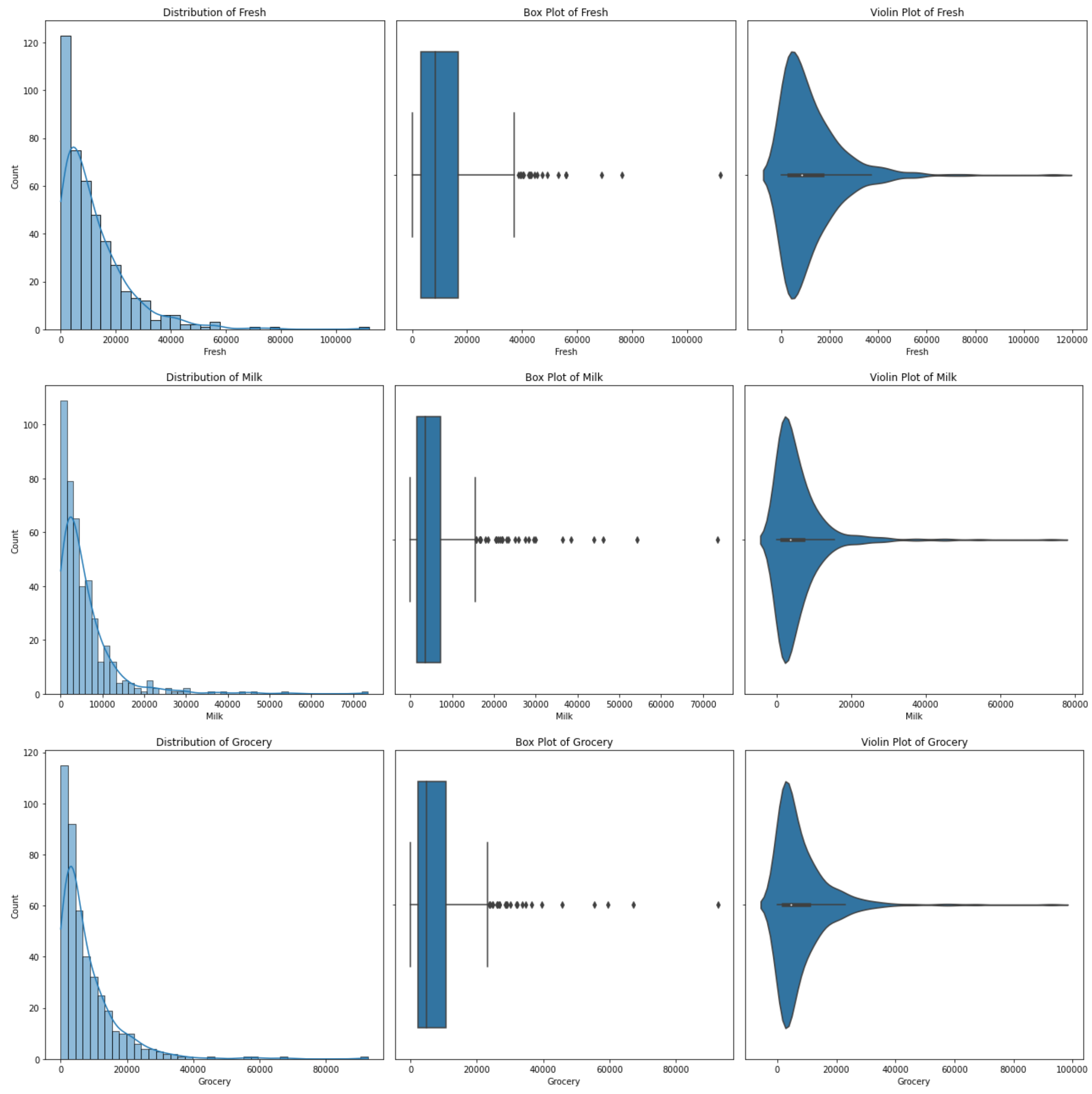
```
Channel
1    54.712120
2    45.276246
dtype: float64
```

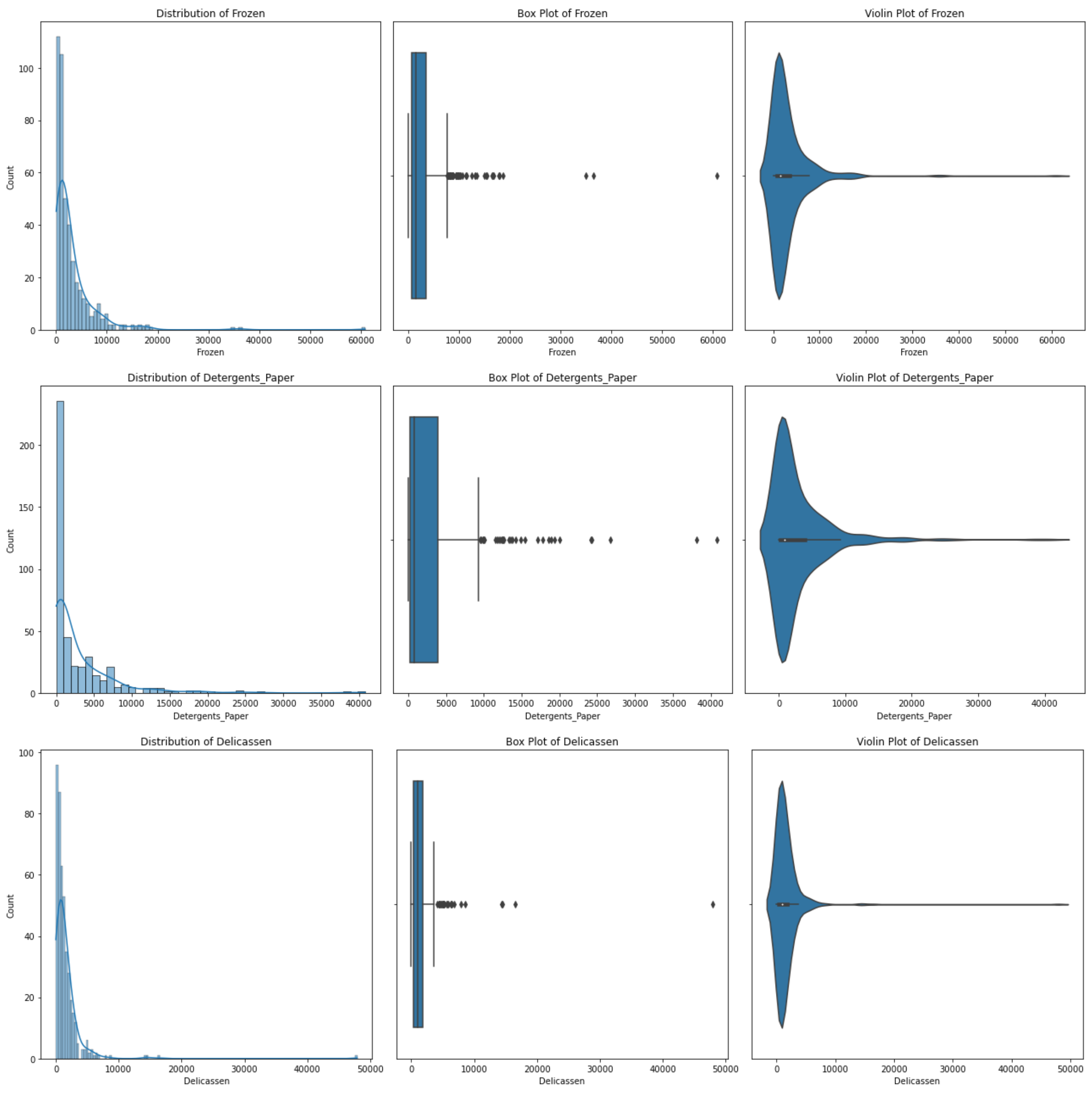
1-6

In [ ]:

```python
# Basic descriptive statistics for each category
print(df.describe().round(2))

# Distribution Analysis with Violin Plots
for column in df.columns[2:]:
    fig, ax = plt.subplots(1, 3, figsize=(18, 6))
    sns.histplot(df[column], kde=True, ax=ax[0])
    ax[0].set_title(f'Distribution of {column}')
    sns.boxplot(x=df[column], ax=ax[1])
    ax[1].set_title(f'Box Plot of {column}')
    sns.violinplot(x=df[column], ax=ax[2])
    ax[2].set_title(f'Violin Plot of {column}')
    plt.tight_layout()
    plt.show()
```

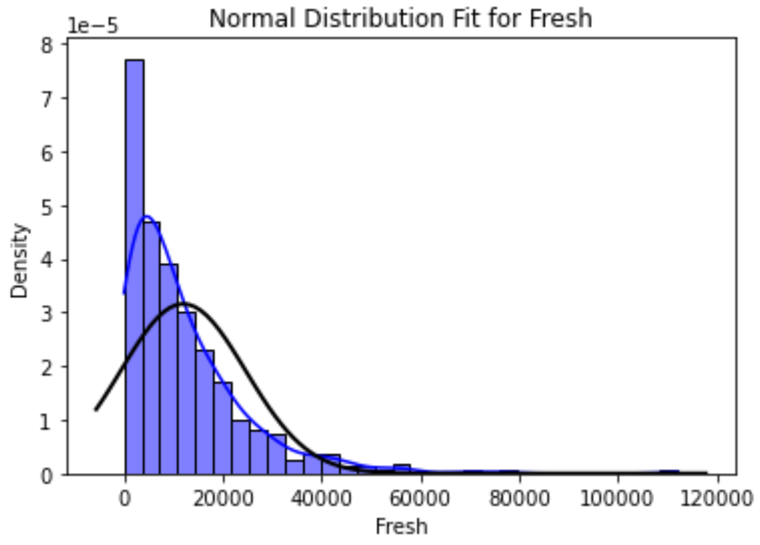
```
       Channel  Region      Fresh      Milk   Grocery    Frozen  \
count   440.00  440.00     440.00    440.00    440.00    440.00
mean      1.32    2.54   12000.30   5796.27   7951.28   3071.93
std       0.47    0.77   12647.33   7380.38   9503.16   4854.67
min       1.00    1.00       3.00     55.00      3.00     25.00
25%       1.00    2.00    3127.75   1533.00   2153.00    742.25
50%       1.00    3.00    8504.00   3627.00   4755.50   1526.00
75%       2.00    3.00   16933.75   7190.25  10655.75   3554.25
max       2.00    3.00  112151.00  73498.00  92780.00  60869.00

       Detergents_Paper  Delicassen
count            440.00      440.00
mean            2881.49     1524.87
std             4767.85     2820.11
min                3.00        3.00
25%              256.75      408.25
50%              816.50      965.50
75%             3922.00     1820.25
max            40827.00    47943.00
```



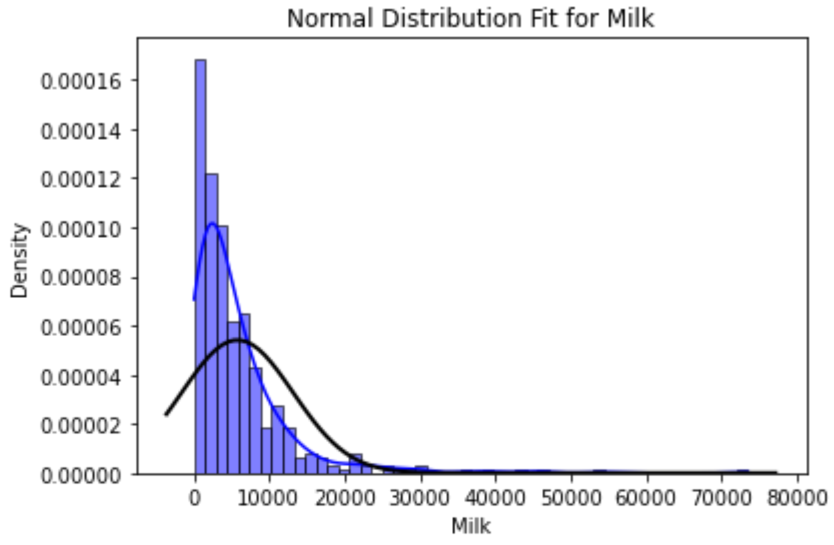

Distribution of Frozen
Box Plot of Frozen
Violin Plot of Frozen
Count
Frozen
Distribution of Detergents_Paper
Box Plot of Detergents_Paper
Violin Plot of Detergents_Paper
Detergents_Paper
Distribution of Delicassen
Box Plot of Delicassen
Violin Plot of Delicassen
Delicassen

1-7

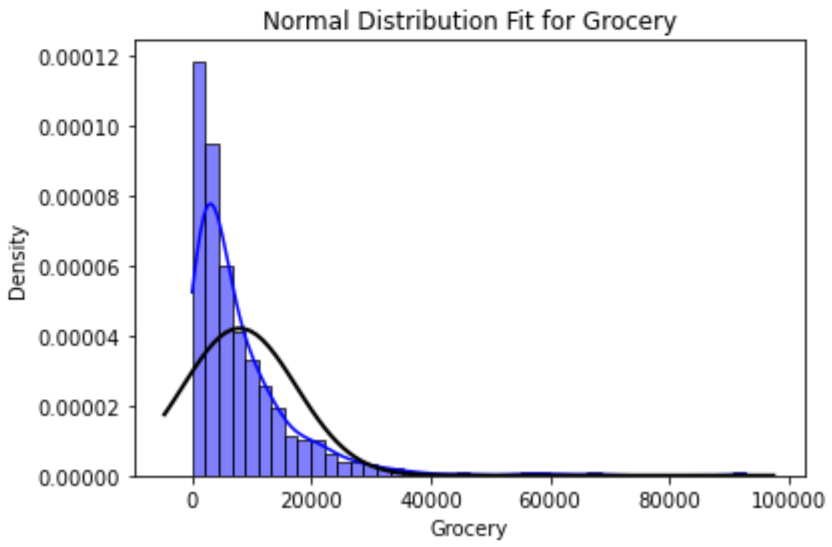
```
In [ ]: # Normal Distribution Fit
        for column in df.columns[2:]:
            mu, std = stats.norm.fit(df[column])
            plt.figure()
            sns.histplot(df[column], kde=True, stat='density', color='blue')
            xmin, xmax = plt.xlim()
            x = np.linspace(xmin, xmax, 100)
            p = stats.norm.pdf(x, mu, std)
            plt.plot(x, p, 'k', linewidth=2)
            plt.title(f'Normal Distribution Fit for {column}')
            plt.xlabel(column)
            plt.ylabel('Density')
            plt.show()
            print(f"Normal Fit for {column}: Mean = {round(mu, 2)}, Std = {round(std, 2)}")

        # Poisson Distribution for 'Frozen'
        lambda_frozen = df['Frozen'].mean()
        print(f"Expected number of purchases (lambda) for 'Frozen': {round(lambda_frozen, 2
```



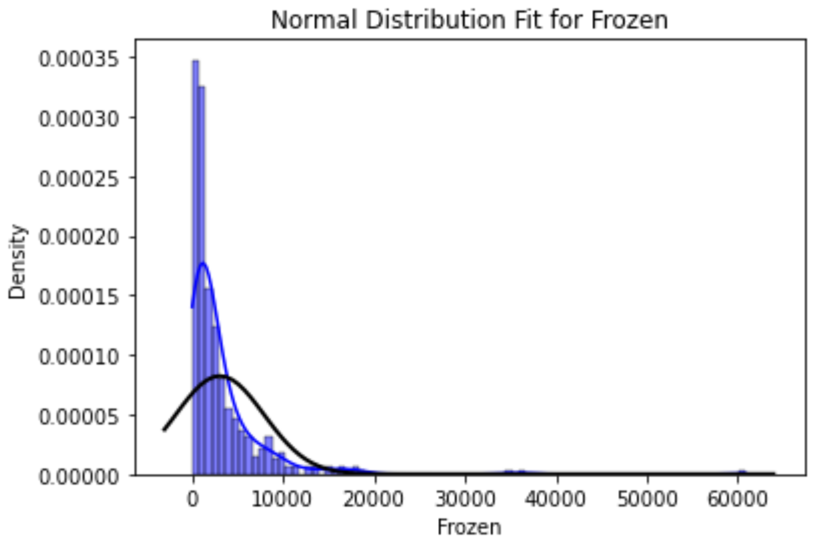
```
Normal Fit for Fresh: Mean = 12000.3, Std = 12632.95
```



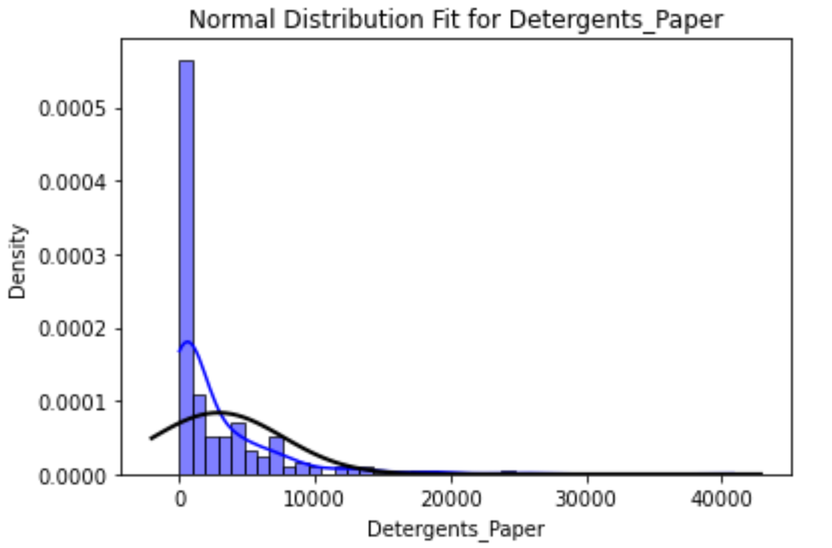
```
Normal Fit for Milk: Mean = 5796.27, Std = 7371.99
```



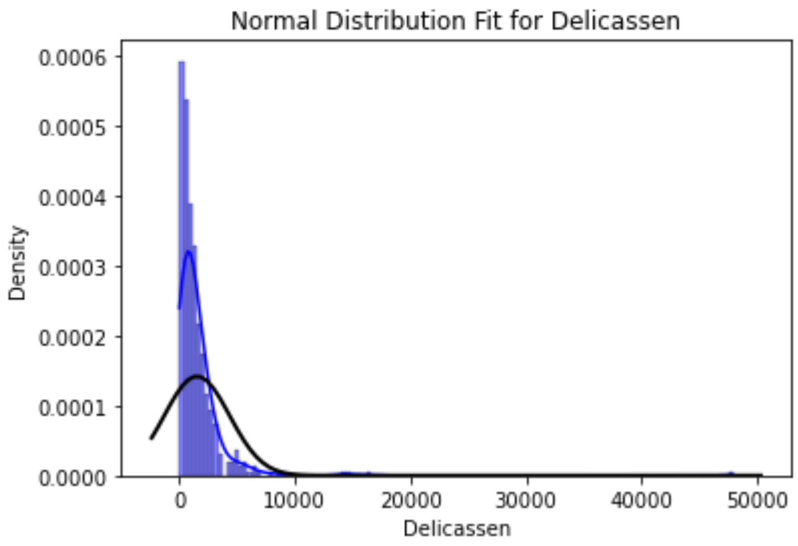
```
Normal Fit for Grocery: Mean = 7951.28, Std = 9492.36
```



```
Normal Fit for Frozen: Mean = 3071.93, Std = 4849.15
```



```
Normal Fit for Detergents_Paper: Mean = 2881.49, Std = 4762.43
```

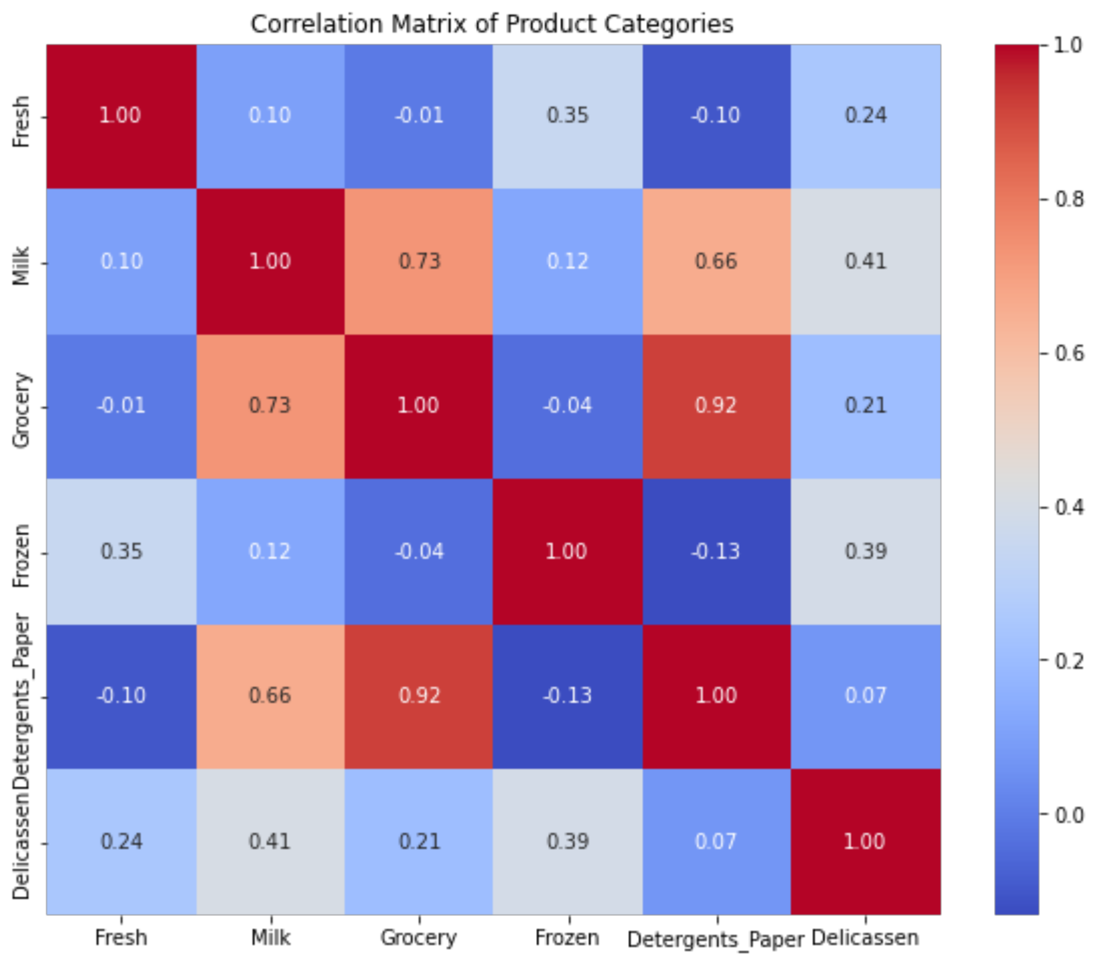


```
Normal Fit for Delicassen: Mean = 1524.87, Std = 2816.9
Expected number of purchases (lambda) for 'Frozen': 3071.93
```

1-8

In [ ]:
```
# Comparing the mean spending on 'Fresh' between two channels
t_stat, p_val = stats.ttest_ind(df[df['Channel'] == 1]['Fresh'], df[df['Channel'] =
print(f"T-test for 'Fresh' between Channels 1 and 2: T-stat={round(t_stat, 2)}, P-v

## Correlations Between Categories
# Heatmap of Pearson correlation coefficients between product categories
plt.figure(figsize=(10, 8))
sns.heatmap(df.iloc[:, 2:].corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix of Product Categories')
plt.show()
```

```
T-test for 'Fresh' between Channels 1 and 2: T-stat=3.59, P-value=0.0
```



1-9

In [ ]:
```
# Drop 'Channel' and 'Region' columns from DataFrame
df_products = df.drop(columns=['Channel', 'Region'], axis=1)
```

1-10

In [ ]:
```
# Display column names of the DataFrame for product categories
print("Columns of the DataFrame for product categories:")
df_products.columns
```

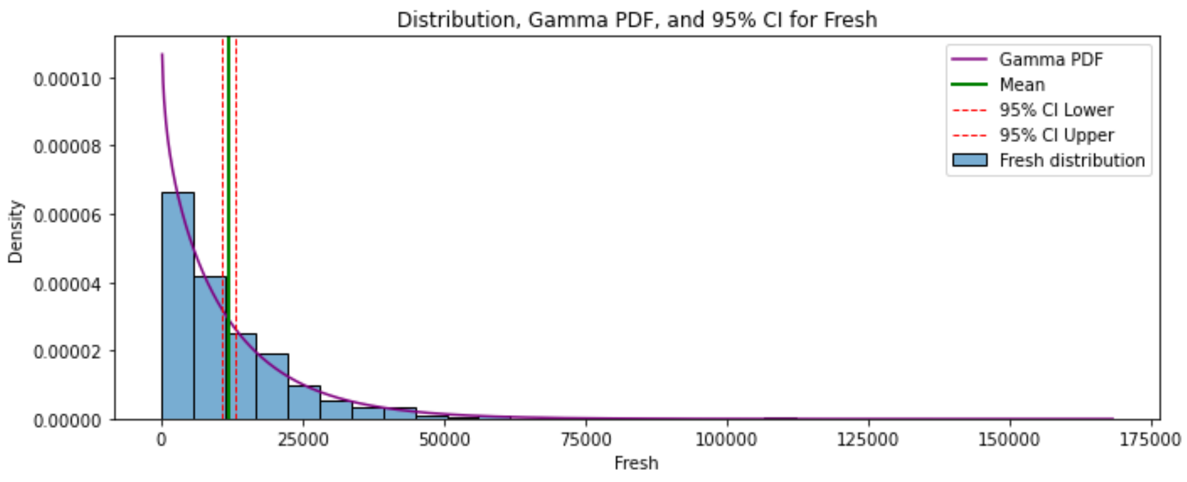
Out[ ]:
```
Columns of the DataFrame for product categories:
Index(['Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper', 'Delicassen'], dty
pe='object')
```

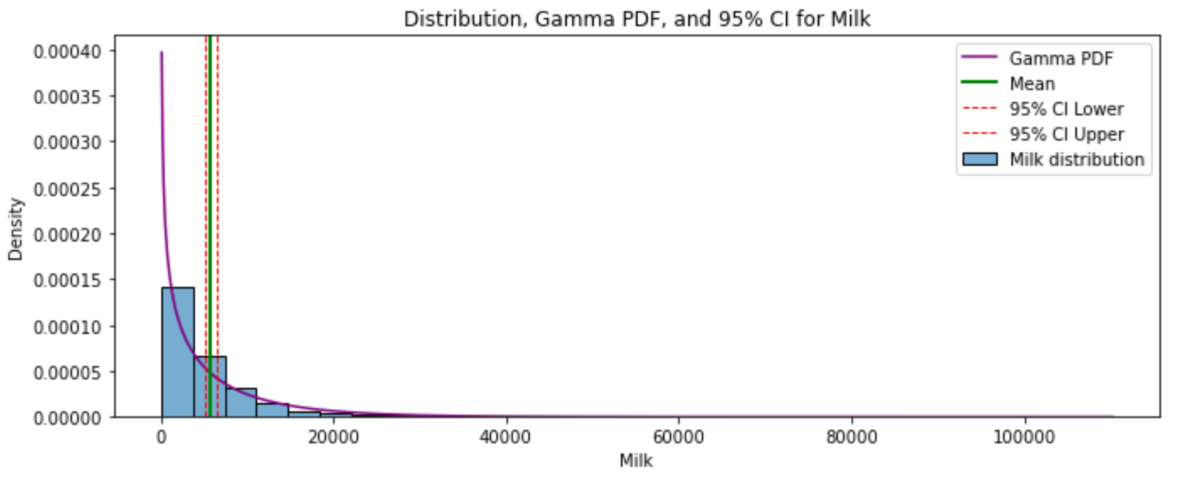
In [ ]:
```
df_products.sum().sort_values(ascending=False)
```

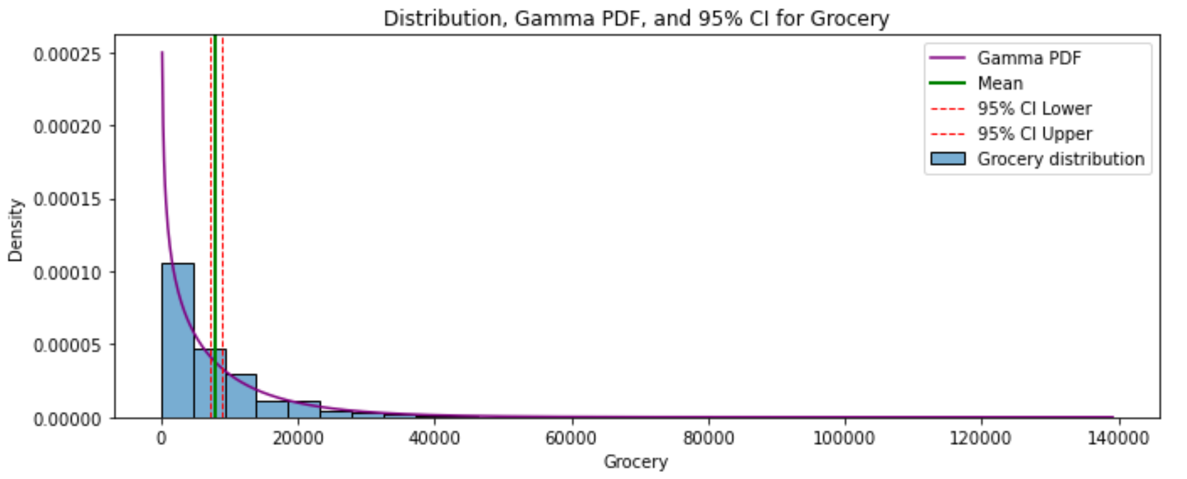
Out[ ]:
```
Fresh               5280131
Grocery             3498562
Milk                2550357
Frozen              1351650
Detergents_Paper    1267857
Delicassen           670943
dtype: int64
```

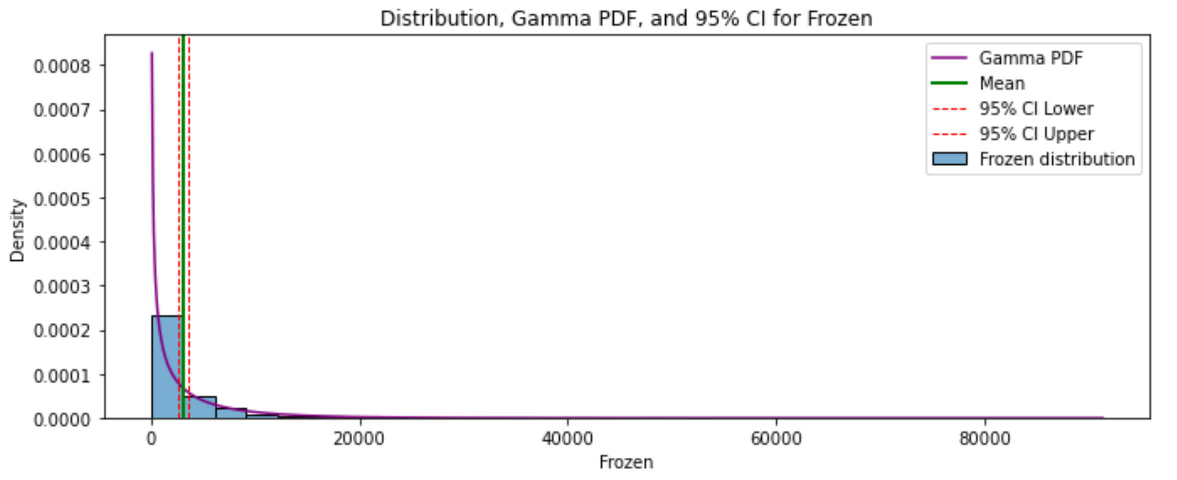
1-11

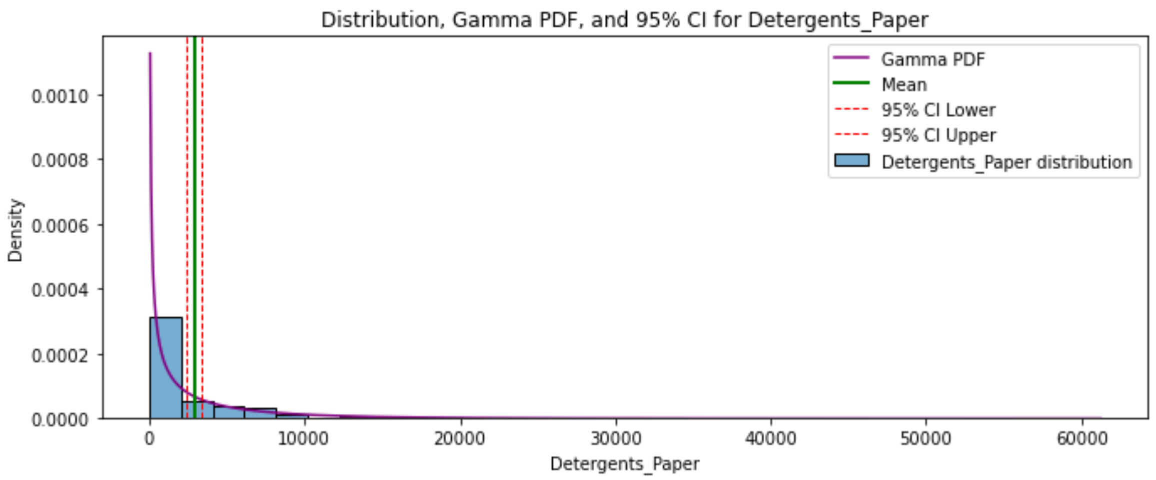
```
In [ ]: # Visualize each column's data distribution, Gamma PDF, and 95% confidence interval

        num_columns = len(df_products.columns)
        fig, axs = plt.subplots(num_columns, 1, figsize=(10, num_columns*4))  # One plot pe

        for idx, column in enumerate(df_products.columns):
            data = df_products[column]
            mean = np.mean(data)
            standard_deviation = np.std(data)
            shape_df = len(data)

            # Calculating the Gamma distribution parameters
            variance_gamma = standard_deviation**2
            k_gamma = mean**2 / variance_gamma  # Shape parameter
            theta_gamma = variance_gamma / mean  # Scale parameter

            # Plotting the histogram of the data
            sns.histplot(data, kde=False, ax=axs[idx], label=f"{column} distribution", alph

            # Overlaying the Gamma PDF
            x = np.linspace(0, max(data)*1.5, 1000)
            y = stats.gamma.pdf(x, a=k_gamma, scale=theta_gamma)
            axs[idx].plot(x, y, label=f"Gamma PDF", color='purple')

            # Calculating and plotting the mean and 95% confidence intervals
            standard_error = standard_deviation / np.sqrt(shape_df)
            lower = mean - 1.96 * standard_error  # For 95% confidence interval
            upper = mean + 1.96 * standard_error
            axs[idx].axvline(x=mean, color='green', linestyle='-', linewidth=2, label='Mean
            axs[idx].axvline(x=lower, color='red', linestyle='--', linewidth=1, label='95%
            axs[idx].axvline(x=upper, color='red', linestyle='--', linewidth=1, label='95%

            axs[idx].set_title(f'Distribution, Gamma PDF, and 95% CI for {column}')
            axs[idx].legend()

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

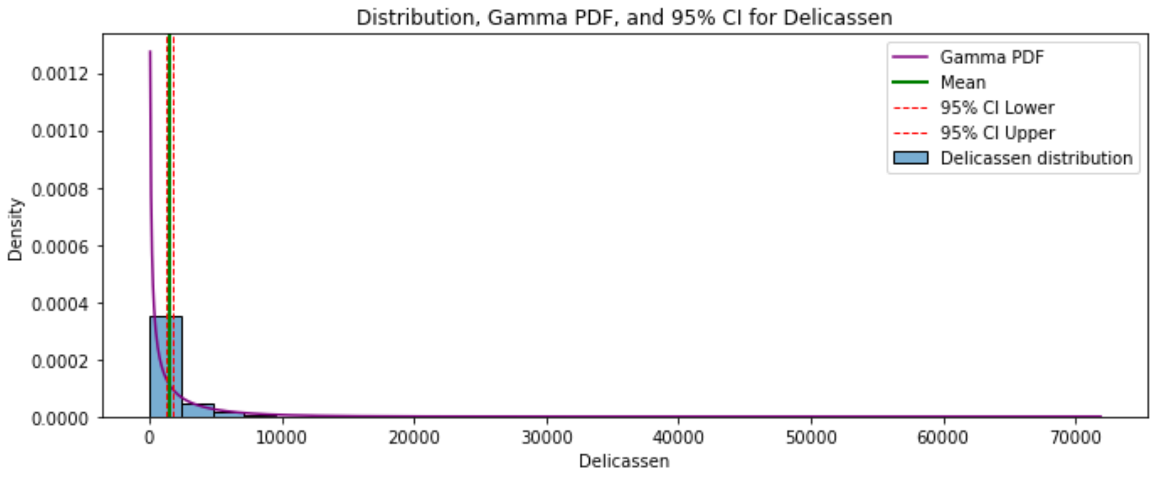

Distribution, Gamma PDF, and 95% CI for Fresh
Gamma PDF
Mean
95% CI Lower
95% CI Upper
Fresh distribution
Density
Fresh
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0
25000
50000
75000
100000
125000
150000
175000


Distribution, Gamma PDF, and 95% CI for Milk
Gamma PDF
Mean
95% CI Lower
95% CI Upper
Milk distribution
Density
Milk
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0.00030
0.00035
0.00040
0
20000
40000
60000
80000
100000


Distribution, Gamma PDF, and 95% CI for Grocery
Gamma PDF
Mean
95% CI Lower
95% CI Upper
Grocery distribution
Density
Grocery
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
0
20000
40000
60000
80000
100000
120000
140000


Distribution, Gamma PDF, and 95% CI for Frozen
Gamma PDF
Mean
95% CI Lower
95% CI Upper
Frozen distribution
Density
Frozen
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0
20000
40000
60000
80000





Testing for Independence

1-13

H_0 : There's no association between Channel sales and Regions

In [ ]:
```
# Define row and column labels
rowlabel = ['Channel 1', 'Channel 2']
collabel = ['Region 1', 'Region 2', 'Region 3']

# Create a cross-tabulation table
table = pd.crosstab(df['Channel'], df['Region'], margins=False, normalize=False)

# Set row and column labels
table.index = rowlabel
table.columns = collabel

# Display the table
print("Cross-tabulation of Channel and Region without normalization:")
print(table)
```

```
Cross-tabulation of Channel and Region without normalization:
           Region 1  Region 2  Region 3
Channel 1        59        28       211
Channel 2        18        19       105
```

In [ ]:
```
# Define row and column labels
rowlabel = ['Channel 1', 'Channel 2']
collabel = ['Region 1', 'Region 2', 'Region 3']

# Create a cross-tabulation table with normalized frequencies
prop = pd.crosstab(df['Channel'], df['Region'], margins=False, normalize=True)

# Set row and column labels
prop.index = rowlabel
prop.columns = collabel

# Display the table with rounded values
print("Cross-tabulation of Channel and Region with normalized frequencies:")
print(prop.round(2))
```

```
Cross-tabulation of Channel and Region with normalized frequencies:
           Region 1  Region 2  Region 3
Channel 1      0.13      0.06      0.48
Channel 2      0.04      0.04      0.24
```

In [ ]:
```
# Convert the previously created 'table' into a
# 'Table' object from statsmodels.
table = sm.stats.Table(table)

# Print the fitted values of the table. Fitted values in the context of a contingen
# are the expected frequencies of observations for each cell of the table under the
# hypothesis of independence between the variables. In other words, it shows what t
# of 'Channel' across 'Region' would look like if there were no association between
print(table.fittedvalues)

# Perform a test for nominal association between 'Channel' and 'Region'. This test
# whether there is a statistically significant association between the two categori
# The test used here is likely a Chi-squared test of independence, which is common
# of analysis. The result of this test includes the Chi-squared statistic and the p
# other details.
X2 = table.test_nominal_association()

print(X2)
```

```
             Region 1   Region 2    Region 3
Channel 1       52.15  31.831818  214.018182
Channel 2       24.85  15.168182  101.981818
df            2
pvalue        0.11365689324243589
statistic     4.349142154535748
```

In [ ]:
```
print("Standardized residuals:")
table.standardized_resids
```

```
Standardized residuals:
```

Out[ ]:

	Region 1	Region 2	Region 3
Channel 1	1.838309	-1.264988	-0.684097
Channel 2	-1.838309	1.264988	0.684097

Based on the results for the Chi-squared test, there is no dependence between the channels and the regions.

p-Val = .099

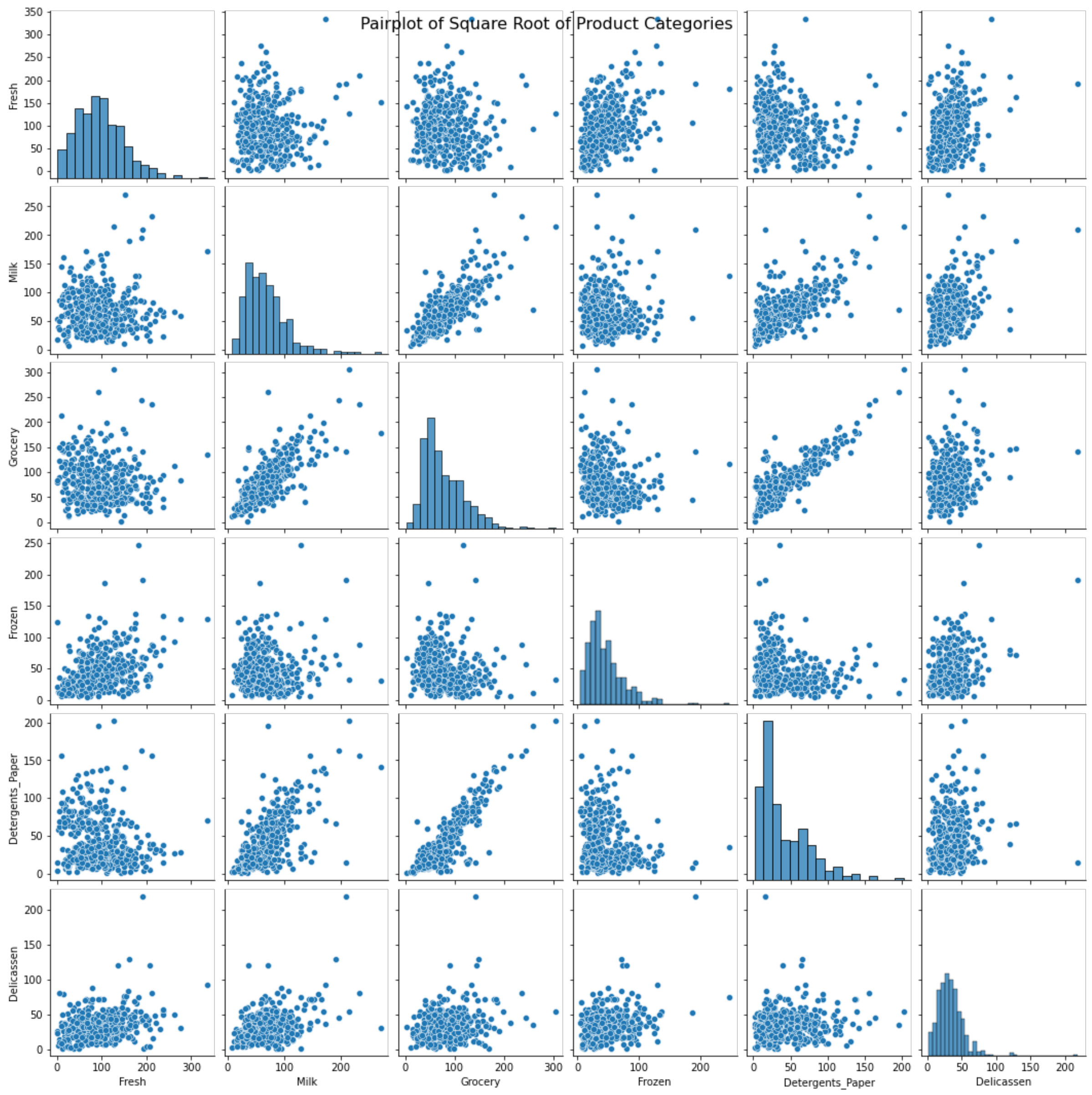
Chi-squared Statistic: .009

Sales Startegy:

- We can execute a sales strategy that is uniform across channel or region. This means that we can design a marketing campaign that is the same across all channels and regions.

1-14

In [ ]:
```
# Plotting pairplot using square root of data
pairplot_fig = sns.pairplot(data=np.sqrt(df_products))
pairplot_fig.fig.suptitle('Pairplot of Square Root of Product Categories', size=16)
plt.show()
```

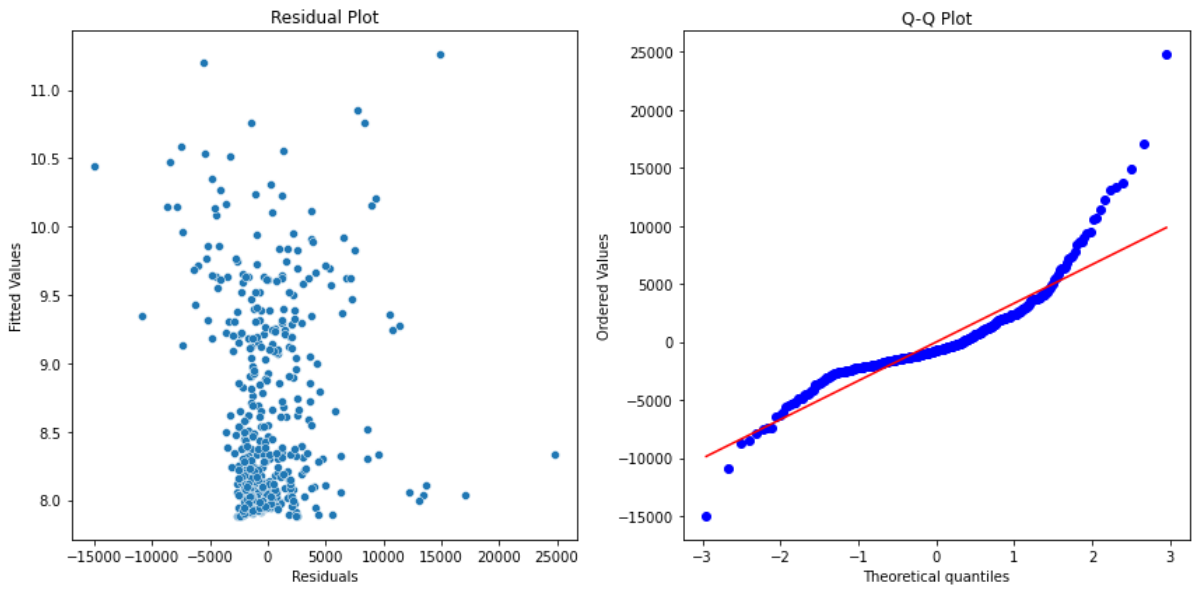


Fitting Models for Prediction

1-15

In [ ]:
```
# Fit Linear Model with interaction term
fitd = smf.ols(formula='Grocery ~ Detergents_Paper', data=df_products).fit()

# Print summary of the model
print("Summary of the Linear Model:")
print(fitd.summary())

print("RMSE:", np.sqrt(mean_squared_error(df['Grocery'], fitd.predict())))
print("MAE:", mean_absolute_error(df['Grocery'], fitd.predict()))

# Diagnostic Plots
fig, axs = plt.subplots(1, 2, figsize=(12, 6))
# Residual Plot
sns.scatterplot(x=fitd.resid, y=np.log(fitd.fittedvalues), ax=axs[0])
axs[0].set_title('Residual Plot')
axs[0].set_xlabel('Residuals')
axs[0].set_ylabel('Fitted Values')
# Q-Q Plot
stats.probplot(fitd.resid, plot=axs[1])
axs[1].set_title('Q-Q Plot')
plt.tight_layout()
plt.show()

# Feature Importance
feature_importance = fitd.params
print("Feature Importance:")
print(feature_importance)
```

```
Summary of the Linear Model:
                            OLS Regression Results
==============================================================================
Dep. Variable:                Grocery   R-squared:                       0.855
Model:                            OLS   Adj. R-squared:                  0.855
Method:                 Least Squares   F-statistic:                     2582.
Date:                Fri, 23 Feb 2024   Prob (F-statistic):          9.56e-186
Time:                        13:56:41   Log-Likelihood:                -4229.2
No. Observations:                 440   AIC:                             8462.
Df Residuals:                     438   BIC:                             8471.
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975
----------------------------------------------------------------------------------
Intercept        2640.7728    201.892     13.080      0.000    2243.976    3037.57
Detergents_Paper    1.8430      0.036     50.812      0.000       1.772       1.91
==============================================================================
Omnibus:                      190.986   Durbin-Watson:                   2.056
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1474.278
Skew:                           1.680   Prob(JB):                         0.00
Kurtosis:                      11.314   Cond. No.                     6.51e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly sp
cified.
[2] The condition number is large, 6.51e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
RMSE: 3615.0755545532143
MAE: 2391.9006173403636
```



```
Feature Importance:
Intercept           2640.772792
Detergents_Paper       1.842970
dtype: float64
```

1-16

In [ ]:
```
# Fitting a generalized linear model (GLM) with gamma distribution
import warnings
from statsmodels.tools.sm_exceptions import DomainWarning

# Suppress DomainWarning
warnings.filterwarnings("ignore", category=DomainWarning)
fitd = smf.glm(formula='Grocery ~ Detergents_Paper + Milk',
               family=sm.families.Gamma(link=sm.families.links.identity()),
               data=df_products).fit()

# Print summary of the model
print("Summary of the GLM with gamma distribution:")
print(fitd.summary())

# Print AIC (Akaike Information Criterion)
print(f"AIC: {fitd.aic}")
```

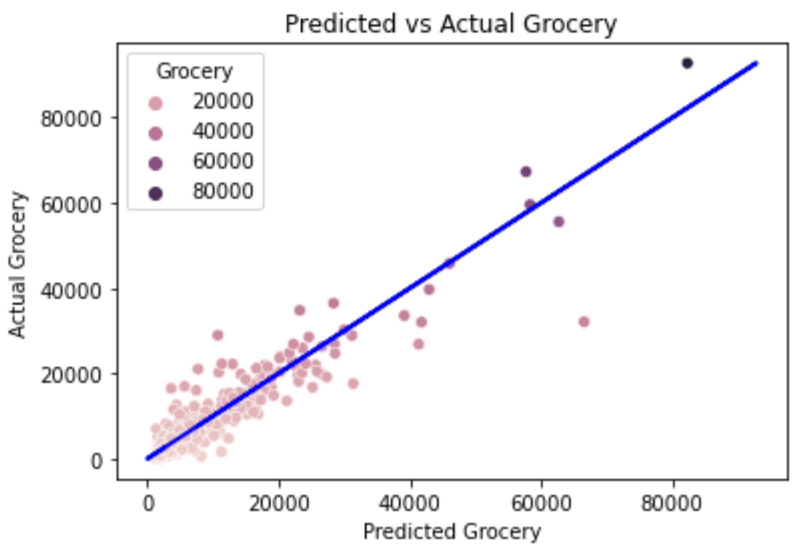
```
Summary of the GLM with gamma distribution:
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                Grocery   No. Observations:                  440
Model:                            GLM   Df Residuals:                      437
Model Family:                   Gamma   Df Model:                            2
Link Function:               identity   Scale:                         0.30160
Method:                          IRLS   Log-Likelihood:                -4071.8
Date:                Fri, 23 Feb 2024   Deviance:                       124.26
Time:                        14:43:20   Pearson chi2:                     132.
No. Iterations:                    13   Pseudo R-squ. (CS):             0.9289
Covariance Type:            nonrobust
================================================================================
                       coef    std err          z      P>|z|      [0.025      0.975
--------------------------------------------------------------------------------
Intercept         1186.4343    113.216     10.479      0.000     964.535    1408.33
Detergents_Paper     1.4182      0.119     11.873      0.000       1.184       1.65
Milk                 0.5022      0.054      9.324      0.000       0.397       0.60
================================================================================
AIC: 8149.556532416005
```

Observation:

Based on the results of our Linear models we see that the result of the test AIC, the Gamma family showed better results in predicting groceries better than the OLS

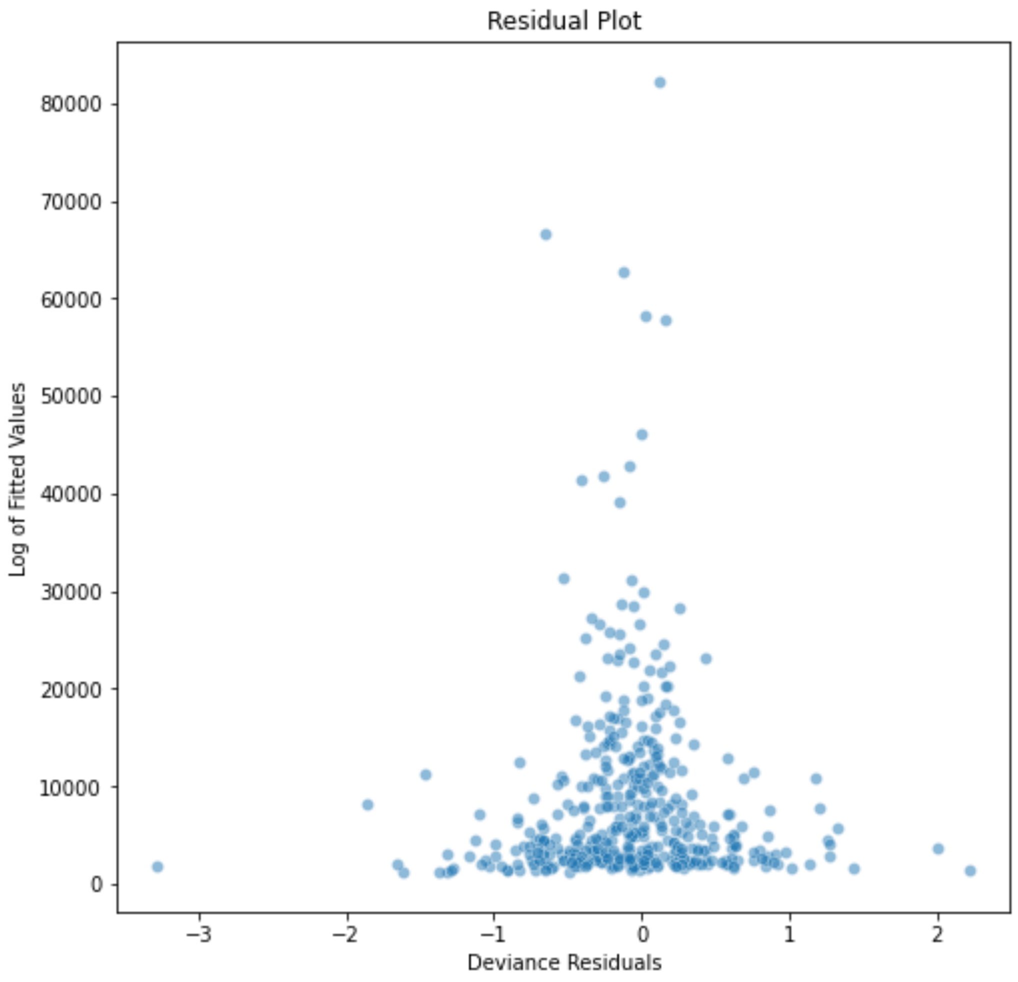
1-17

In [ ]:
```
# Checking predicted values against actual points
sns.scatterplot(x=fitd.predict(), y=df_products['Grocery'], hue=df_products['Grocer
# Line for perfect predictions
plt.plot(df['Grocery'], df['Grocery'], color='blue', linewidth=2)
plt.title('Predicted vs Actual Grocery')
plt.xlabel('Predicted Grocery')
plt.ylabel('Actual Grocery')
plt.show()
```



1-18

In [ ]:
```
# Plotting residual plot
plt.figure(figsize=(8,8))
sns.scatterplot(x=fitd.resid_deviance, y=fitd.fittedvalues, alpha =0.50)
plt.title('Residual Plot')
plt.xlabel('Deviance Residuals')
plt.ylabel('Log of Fitted Values')
plt.show()
```



Forecasting Equation:

$$groceries = 1186 + 1.42(Detergents/paper) + .50(Milk) MonetaryUnits$$