Experiment No. 5
Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset
Date of Performance: 21 / 08 /2023
Date of Submission: 03 / 09 /2023

**Aim:** Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset.

Objective: Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

Theory:

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For example: The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.
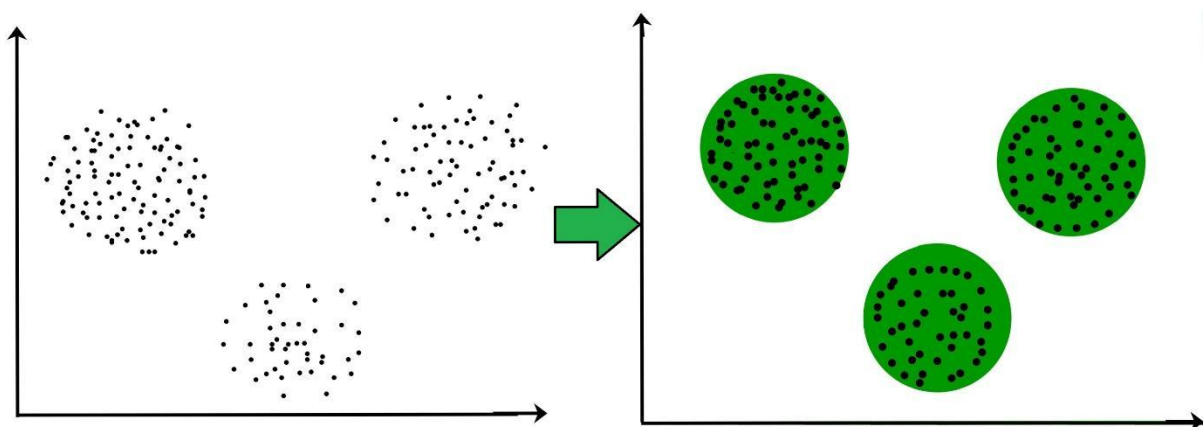
Dataset:

This data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories. The wholesale distributor operating in different regions of Portugal has information on annual spending of several items in their stores across different regions and channels. The dataset consist of 440 large retailers annual spending on 6 different varieties of product in 3 different regions (lisbon , oporto, other) and across different sales channel (Hotel, channel)

Detailed overview of dataset

Records in the dataset = 440 ROWS

Columns in the dataset = 8 COLUMNS

FRESH: annual spending (m.u.) on fresh products (Continuous)

MILK:- annual spending (m.u.) on milk products (Continuous)

GROCERY:- annual spending (m.u.) on grocery products (Continuous)

FROZEN:- annual spending (m.u.) on frozen products (Continuous)

DETERGENTS_PAPER :- annual spending (m.u.) on detergents and paper products (Continuous)

DELICATESSEN:- annual spending (m.u.)on and delicatessen products (Continuous);

CHANNEL: - sales channel Hotel and Retailer

REGION:- three regions (Lisbon, Oporto, Other)

Code:

ml-exp5

September 4, 2023

```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```python
import pandas as pd

# Define a function to load the data
def load_data(path):
    try:
        df = pd.read_csv(path)
        print("Data loaded successfully!")
        return df
    except Exception as e:
        print(f"An error occurred: {e}")
        return None

# Path to the data file
path = '/content/Wholesale customers data.csv'

# Load the data
df = load_data(path)

# Display the first few rows of the DataFrame
print(df.head())
```

```
Data loaded successfully!
   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
```

```python
print("Column names:")
print(df.columns)
```

Column names:
Index(['Channel', 'Region', 'Fresh', 'Milk', 'Grocery', 'Frozen',
       'Detergents_Paper', 'Delicassen'],
      dtype='object')

```python
# Print the data types of each column
print("Data types:")
print(df.dtypes)
```

Data types:
Channel   int64
Region   int64
Fresh   int64
Milk   int64
Grocery   int64
Frozen   int64
Detergents_Paper   int64
Delicassen   int64
dtype: object

```python
# Check for missing values
print("Missing values per column:")
print(df.isnull().sum())
```

Missing values per column:
Channel   0
Region   0
Fresh   0
Milk   0
Grocery   0
Frozen   0
Detergents_Paper   0
Delicassen   0
dtype: int64

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

# Check descriptive statistics
print("Descriptive Statistics:")
print(df.describe())

# Check for duplicates
print("Number of duplicate rows: ", df.duplicated().sum())
```
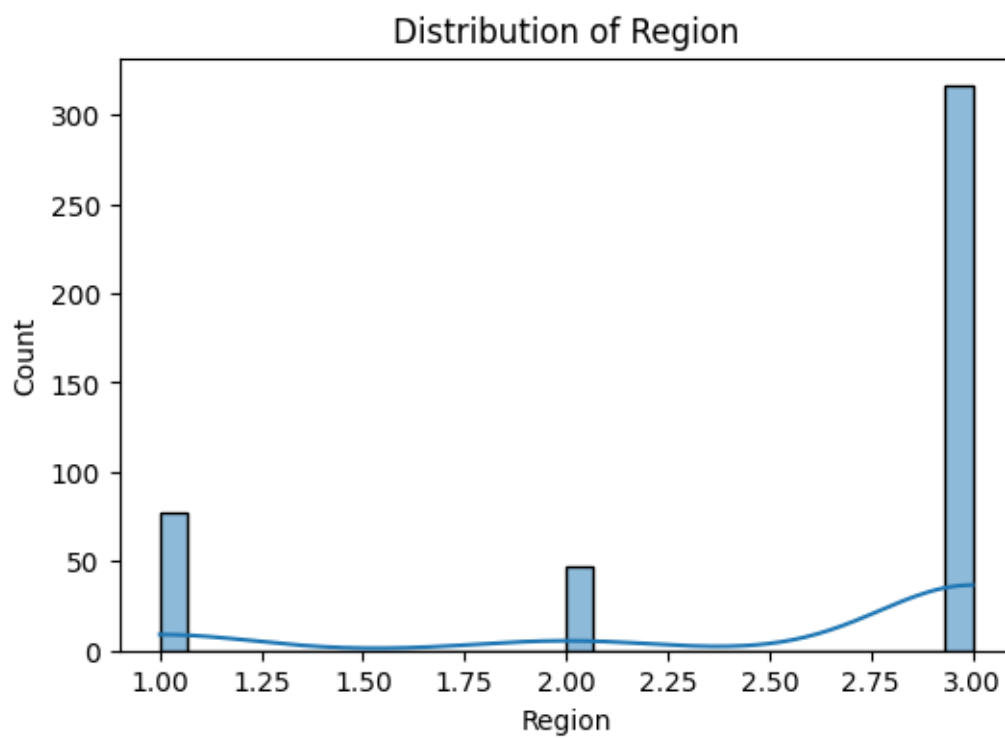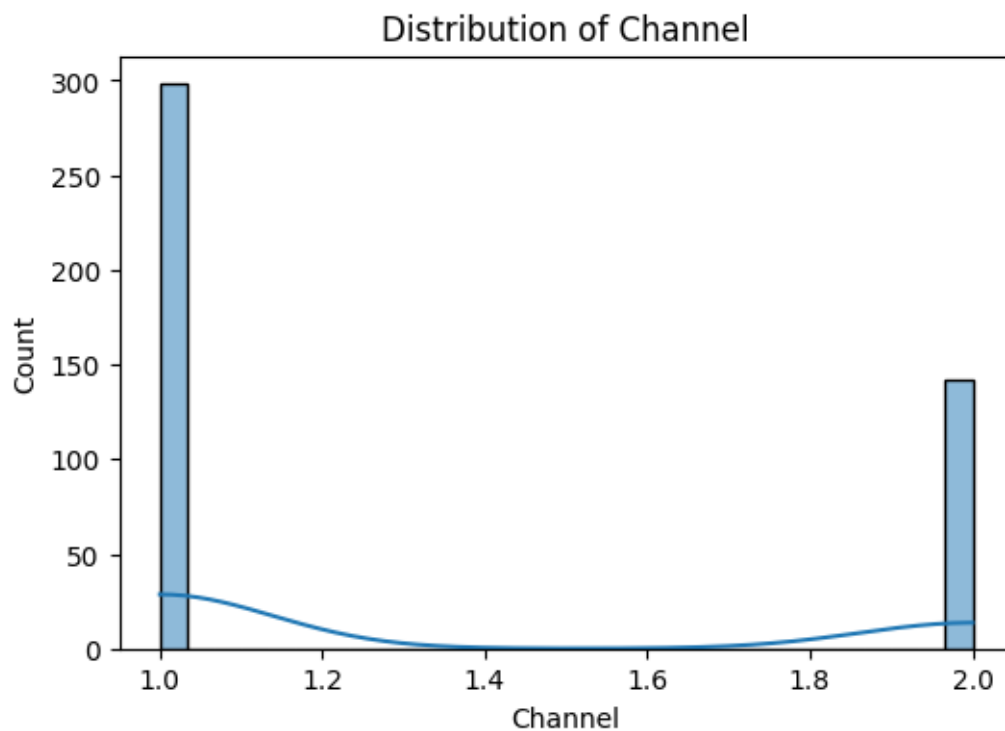
```python
# Distribution plots for each feature
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[column], bins=30, kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()
```
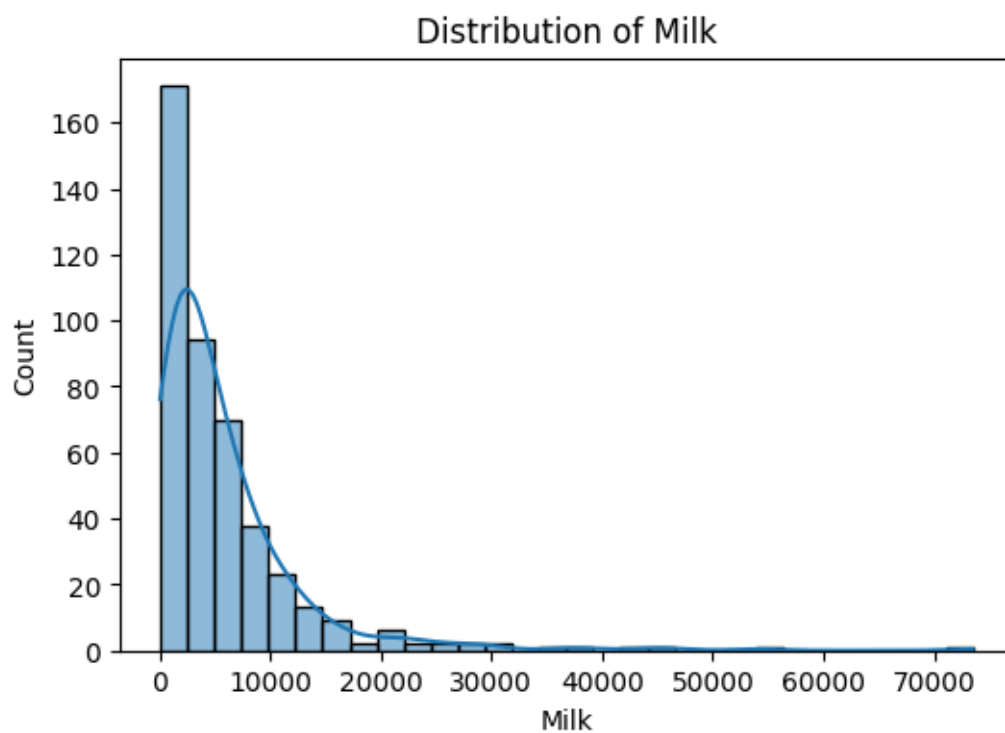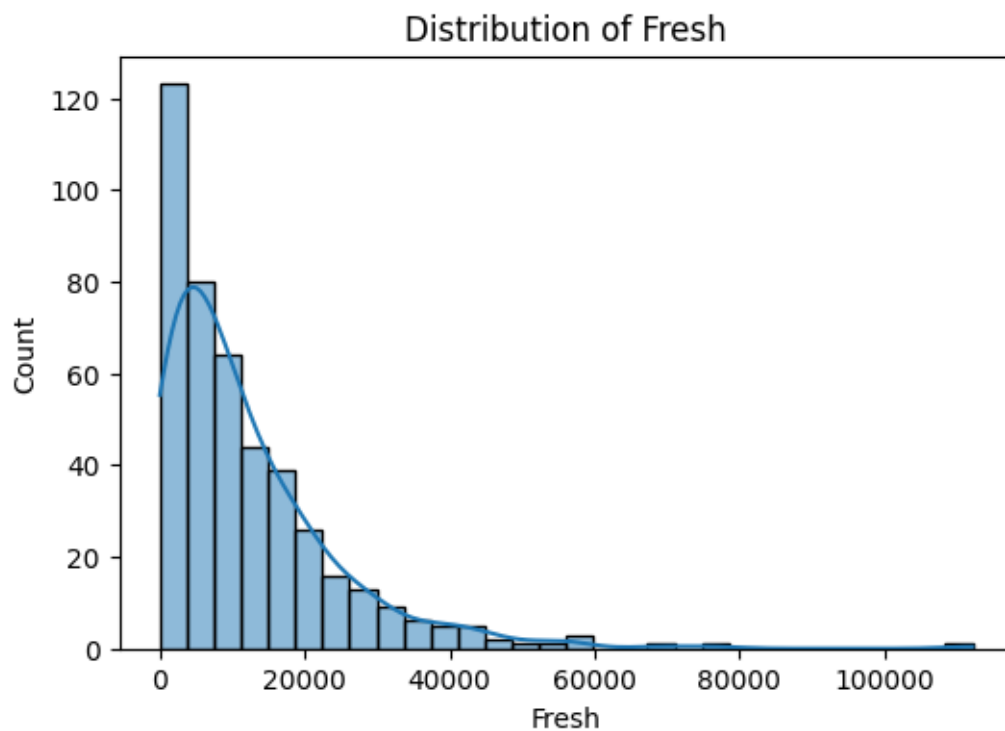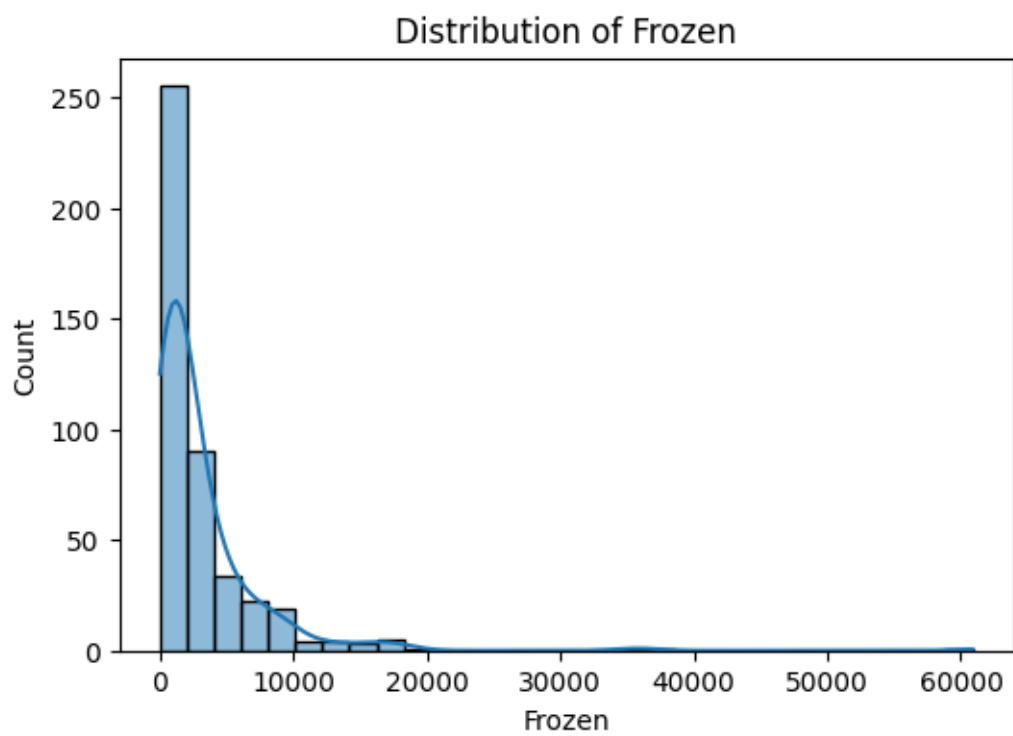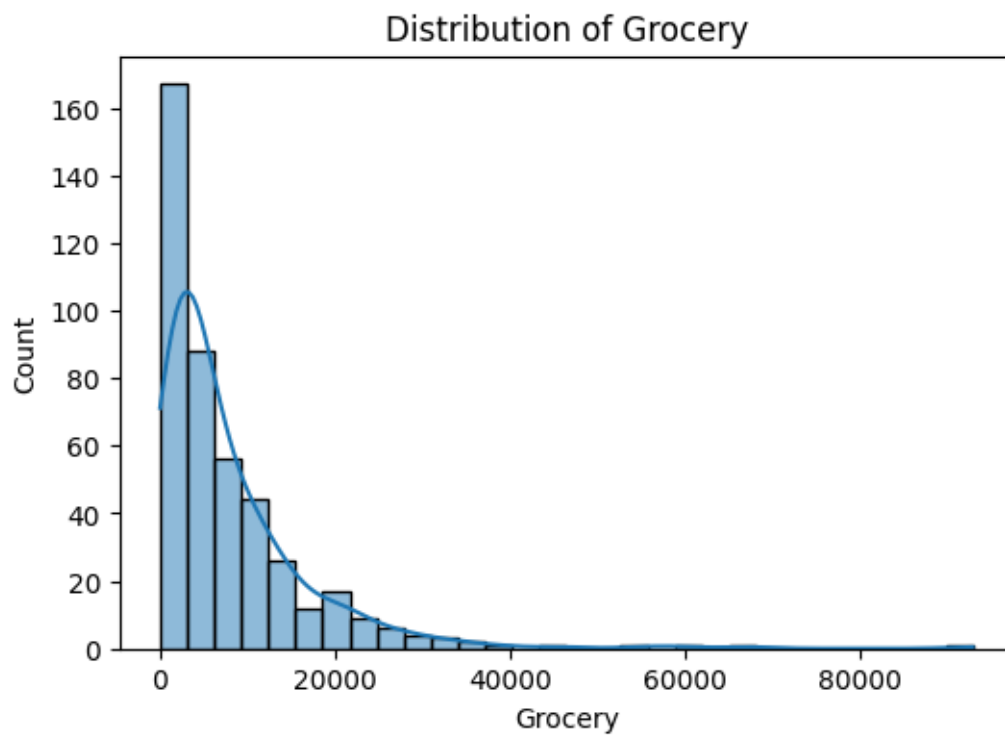
Descriptive Statistics:

	Channel	Region	Fresh	Milk	Grocery
count	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829
min	1.000000	1.000000	3.000000	55.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000

	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000
mean	3071.931818	2881.493182	1524.870455
std	4854.673333	4767.854448	2820.105937
min	25.000000	3.000000	3.000000
25%	742.250000	256.750000	408.250000
50%	1526.000000	816.500000	965.500000
75%	3554.250000	3922.000000	1820.250000
max	60869.000000	40827.000000	47943.000000

Number of duplicate rows: 0
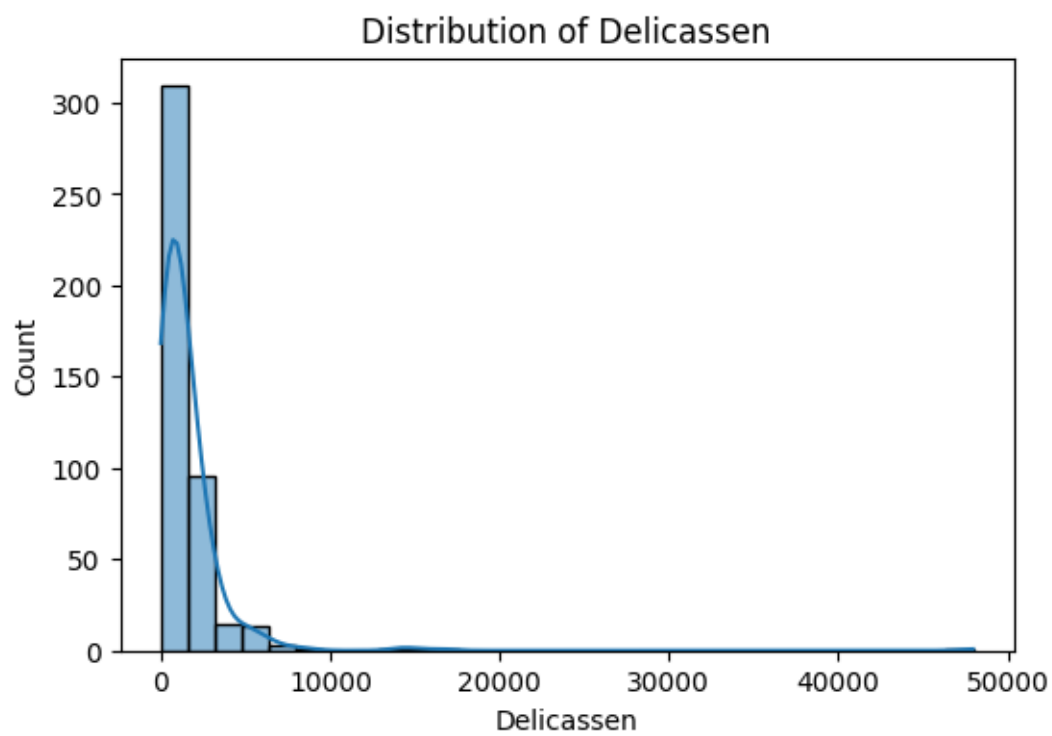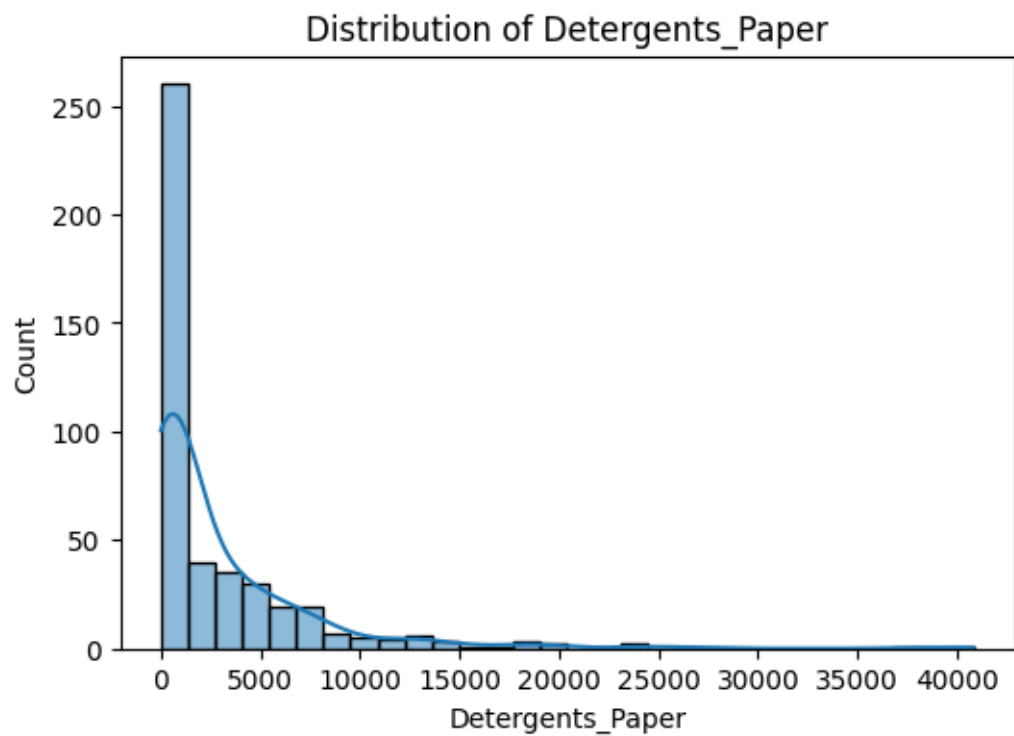
Distribution of Channel

Distribution of Region

Distribution of Fresh

Distribution of Milk

Distribution of Grocery

Distribution of Frozen

Distribution of Detergents_Paper

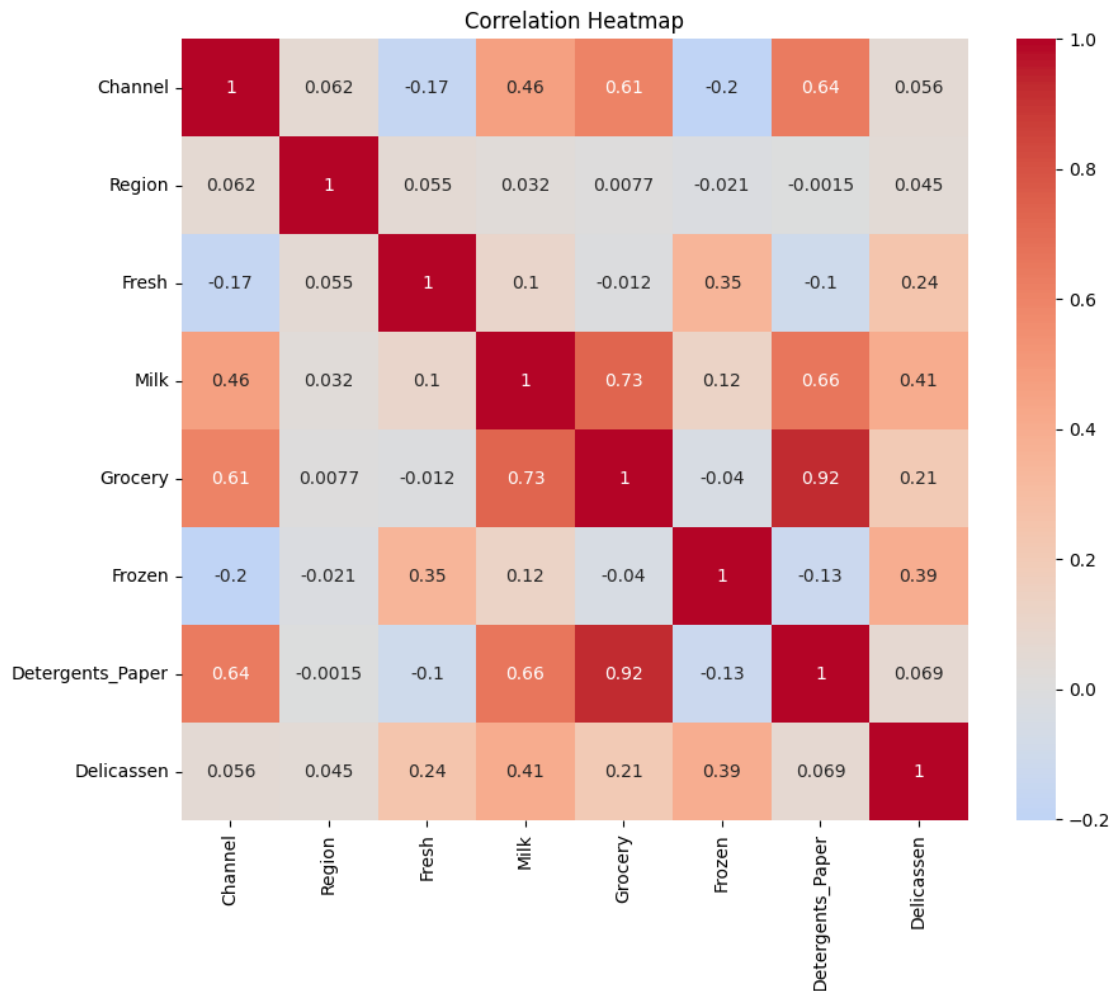# Distribution of Delicassen

```
[ ]: # Heatmap for correlation between variables
     plt.figure(figsize=(10, 8))
     sns.heatmap(df.corr(), annot=True, cmap='coolwarm', center=0)
     plt.title('Correlation Heatmap')
     plt.show()
```


Correlation Heatmap

```
[ ]: # checking for outliers
     import seaborn as sns
     import matplotlib.pyplot as plt

     # Draw boxplots for all features
     for column in df.columns:
         plt.figure(figsize=(6, 4))
         sns.boxplot(df[column])
         plt.title(f'Boxplot of {column}')
         plt.show()
```
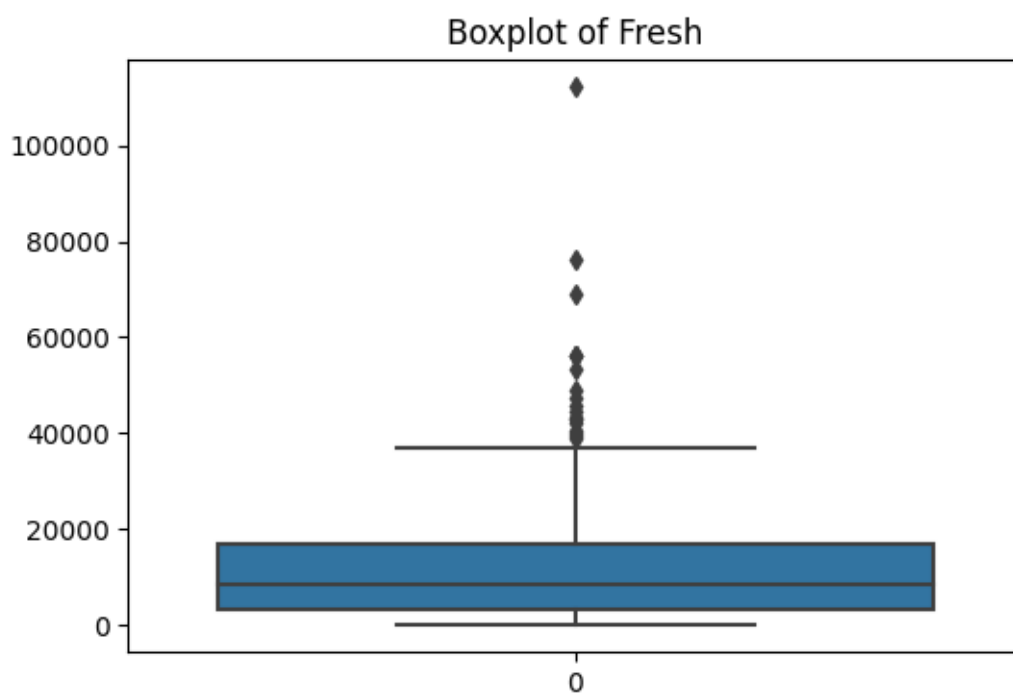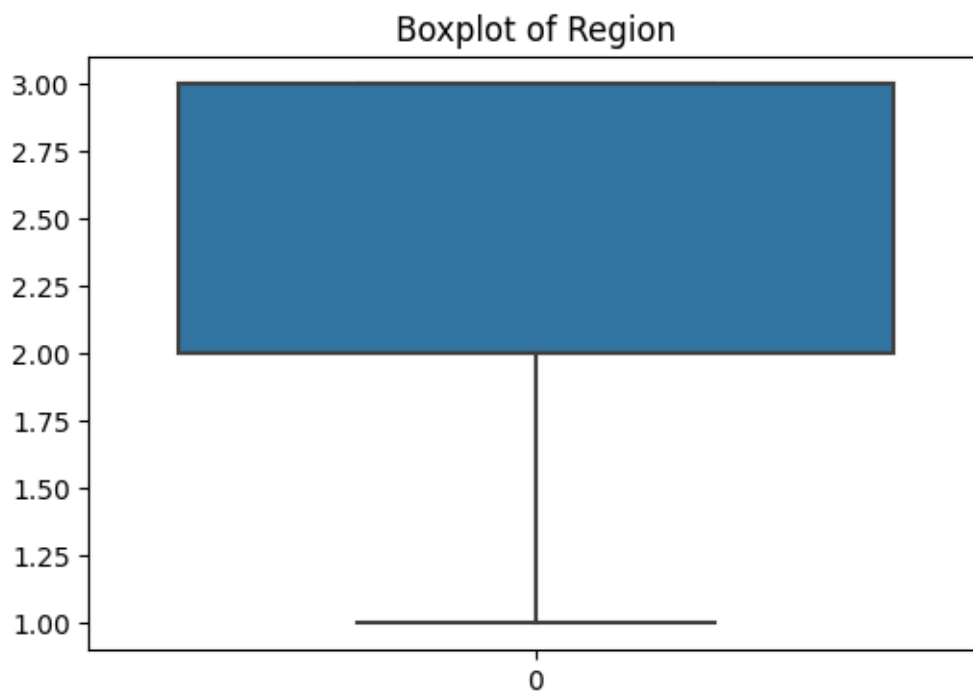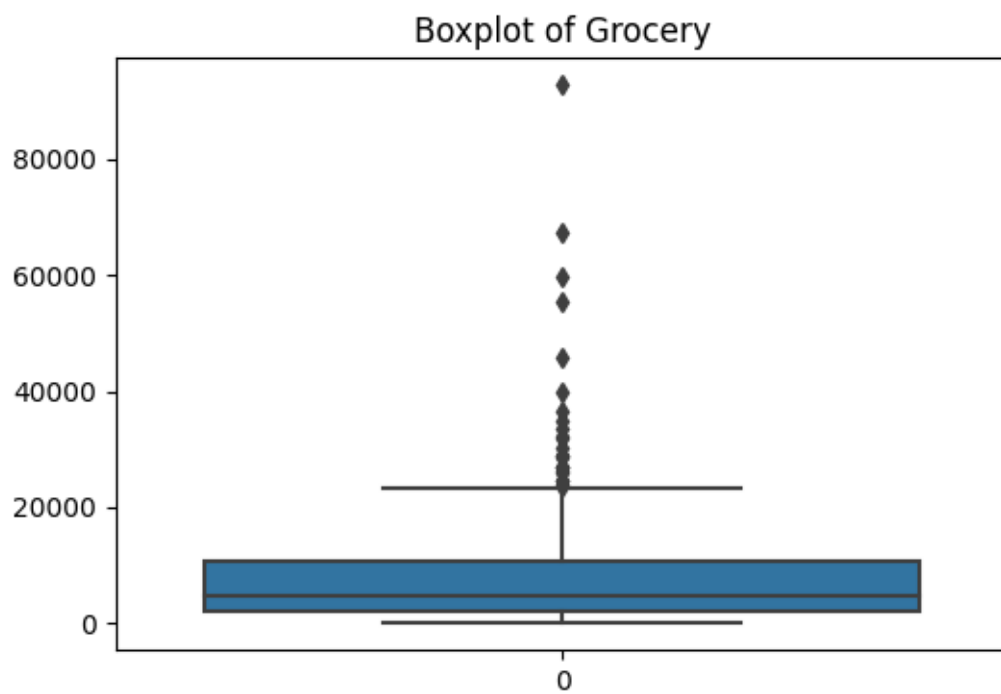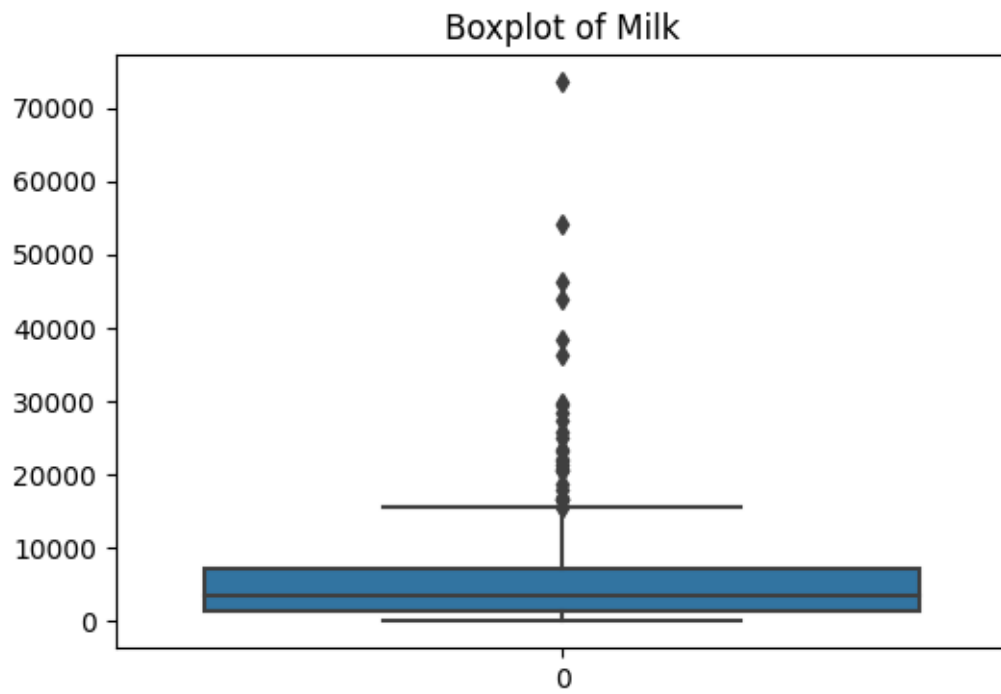
8

```python
# Function to detect outliers
def detect_outliers(dataframe, column):
    Q1 = dataframe[column].quantile(0.25)
    Q3 = dataframe[column].quantile(0.75)
    IQR = Q3 - Q1
    outliers = dataframe[(dataframe[column] < Q1 - 1.5*IQR) |↵
 ↪(dataframe[column] > Q3 + 1.5*IQR)]
    return outliers

# Detect and print number of outliers for each feature
for column in df.columns:
    outliers = detect_outliers(df, column)
    print(f'Number of outliers in {column}: {len(outliers)}')
```
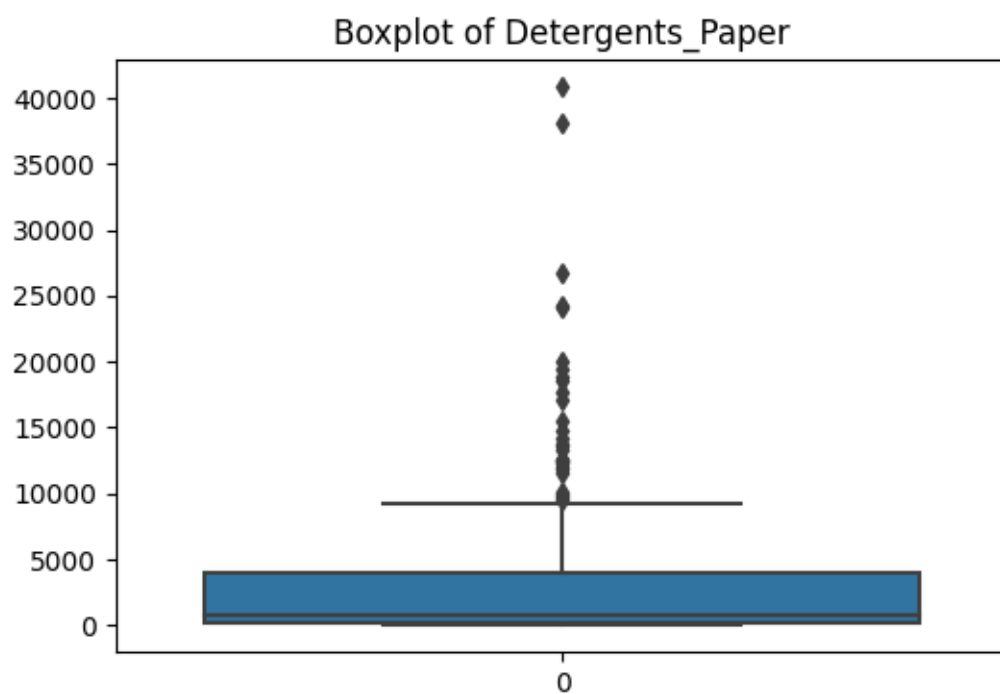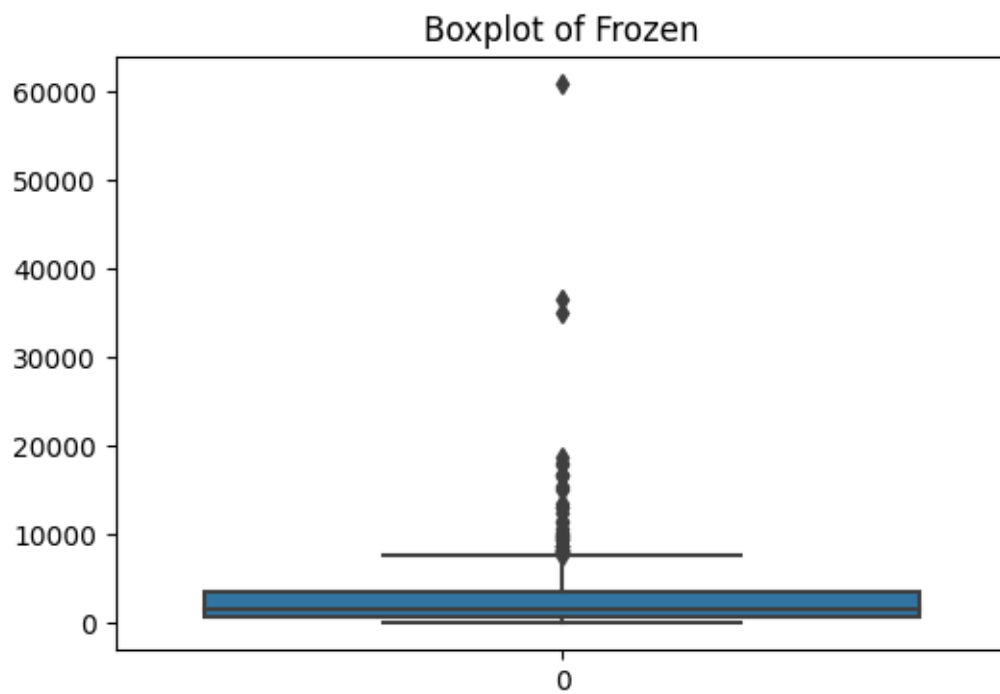

Boxplot of Channel

Boxplot of Region

Boxplot of Fresh

Boxplot of Milk

Boxplot of Grocery

Boxplot of Frozen

Boxplot of Detergents_Paper
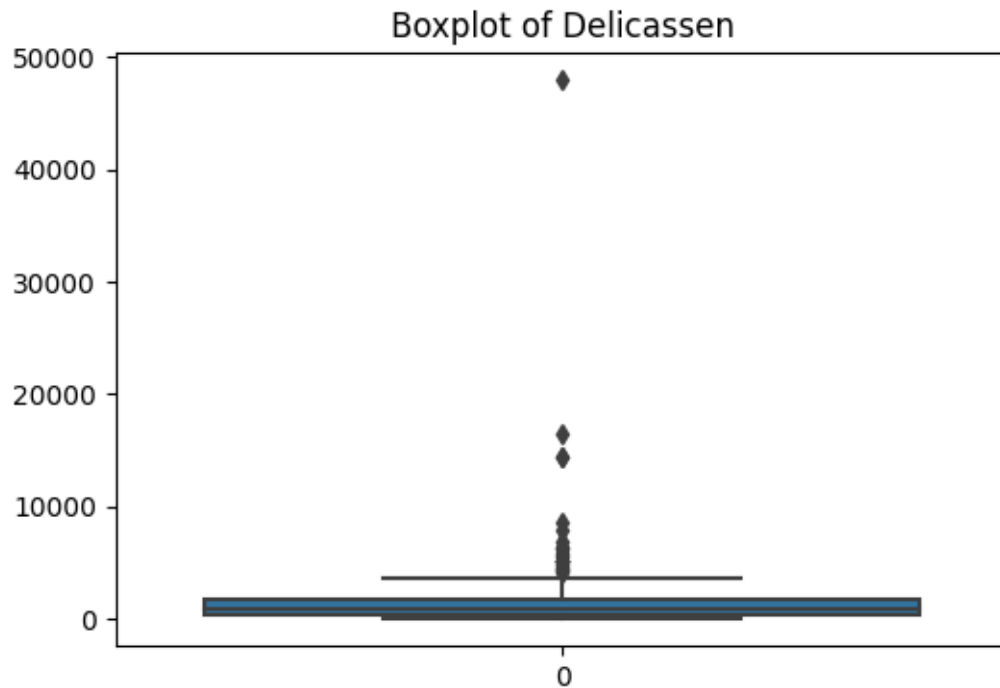
Boxplot of Delicassen


```
Number of outliers in Channel: 0
Number of outliers in Region: 0
Number of outliers in Fresh: 20
Number of outliers in Milk: 28
Number of outliers in Grocery: 24
Number of outliers in Frozen: 43
Number of outliers in Detergents_Paper: 30
Number of outliers in Delicassen: 27
```

```python
def handle_outliers(dataframe, column):
    Q1 = dataframe[column].quantile(0.25)
    Q3 = dataframe[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_limit = Q1 - 1.5*IQR
    upper_limit = Q3 + 1.5*IQR
    dataframe[column] = dataframe[column].apply(lambda x: upper_limit if x >
    ↪upper_limit else lower_limit if x < lower_limit else x)

# Handle outliers for each feature
for column in df.columns:
    handle_outliers(df, column)
```

```python
# Import necessary libraries
import seaborn as sns
```
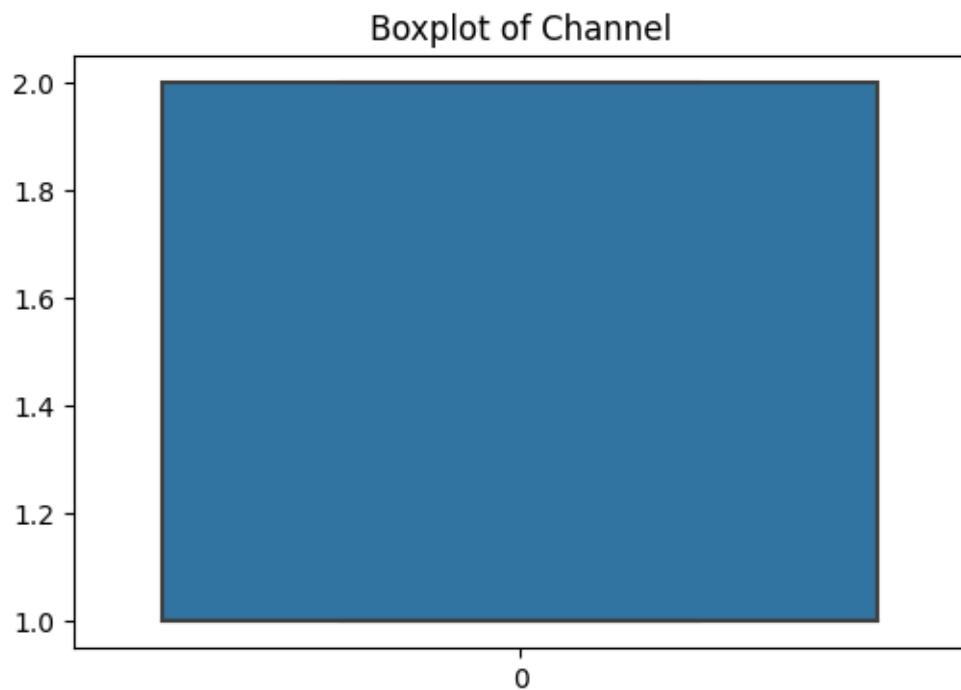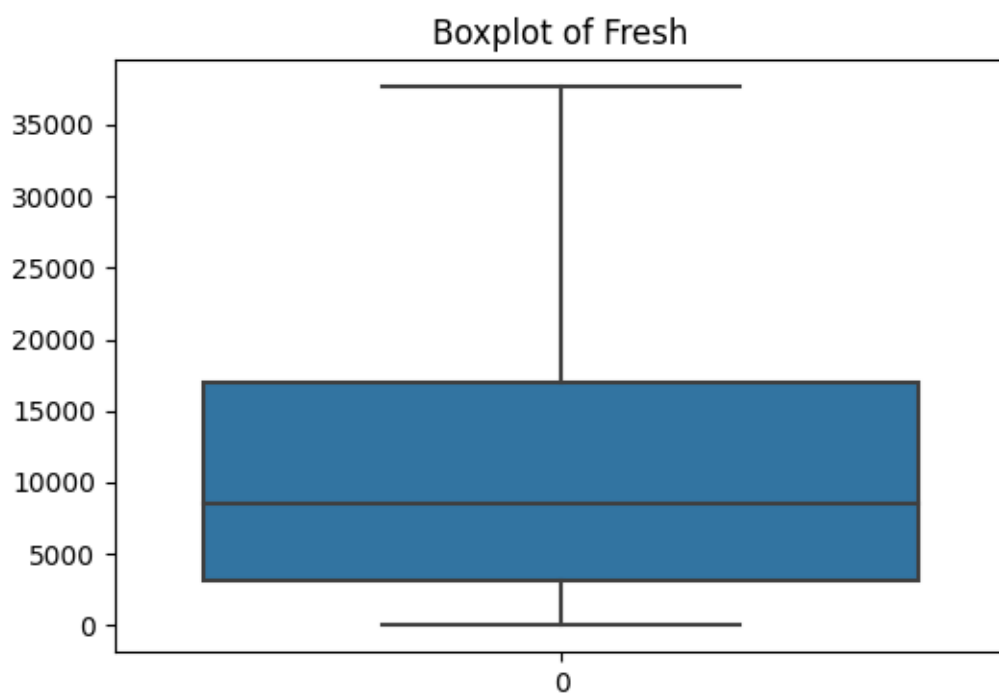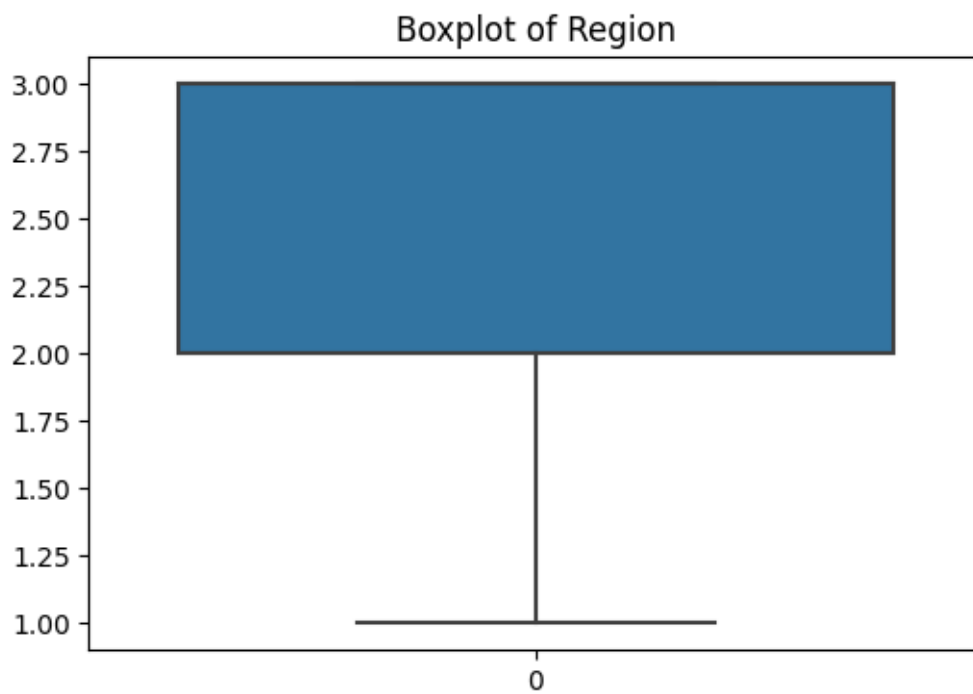
```python
import matplotlib.pyplot as plt

# Draw boxplots for all features
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.boxplot(df[column])
    plt.title(f'Boxplot of {column}')
    plt.show()

# Draw distribution plots for all features
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[column], bins=30, kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()
```


Boxplot of Channel

Boxplot of Region

Boxplot of Fresh

Boxplot of Milk

Boxplot of Grocery

Boxplot of Frozen

Boxplot of Detergents_Paper

Boxplot of Delicassen

Distribution of Channel

Distribution of Region

Distribution of Fresh

Distribution of Milk

Distribution of Grocery

Distribution of Frozen

Distribution of Detergents_Paper

Distribution of Delicassen


```
# Function to detect outliers
def detect_outliers(dataframe, column):
    Q1 = dataframe[column].quantile(0.25)
    Q3 = dataframe[column].quantile(0.75)
    IQR = Q3 - Q1
    outliers = dataframe[(dataframe[column] < Q1 - 1.5*IQR) |
  ↪(dataframe[column] > Q3 + 1.5*IQR)]
    return outliers

# Detect and print number of outliers for each feature
for column in df.columns:
    outliers = detect_outliers(df, column)
    print(f'Number of outliers in {column}: {len(outliers)}')
```

```
Number of outliers in Channel: 0
Number of outliers in Region: 0
Number of outliers in Fresh: 0
Number of outliers in Milk: 0
Number of outliers in Grocery: 0
Number of outliers in Frozen: 0
Number of outliers in Detergents_Paper: 0
Number of outliers in Delicassen: 0
```

```
# Check descriptive statistics
print("Descriptive Statistics:")
print(df.describe())

# Check for duplicates
print("Number of duplicate rows: ", df.duplicated().sum())

# Distribution plots for each feature
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[column], bins=30, kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()
```
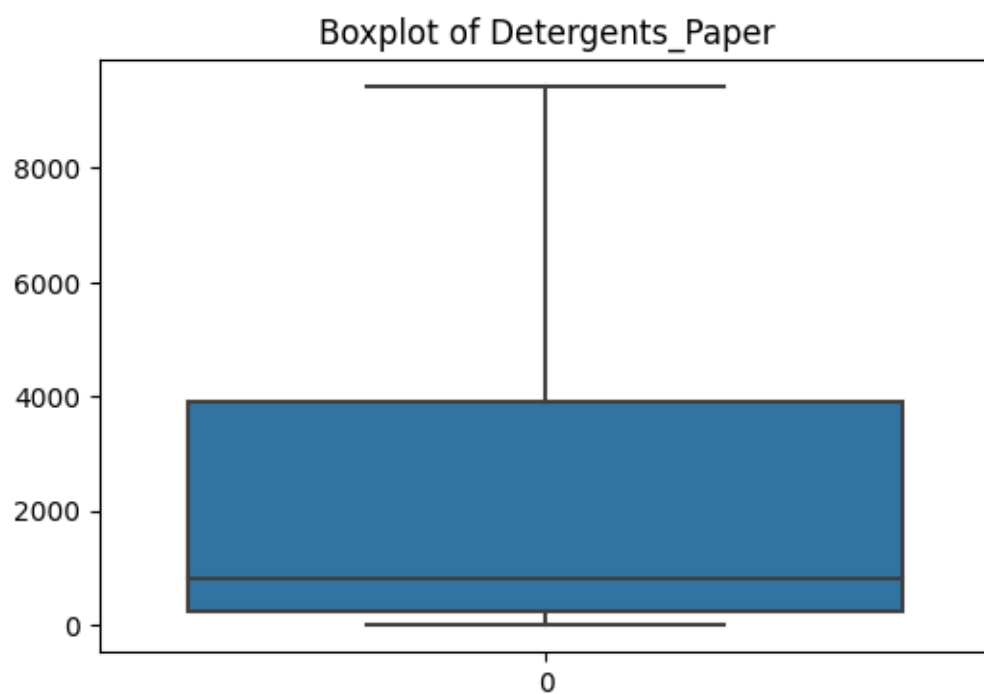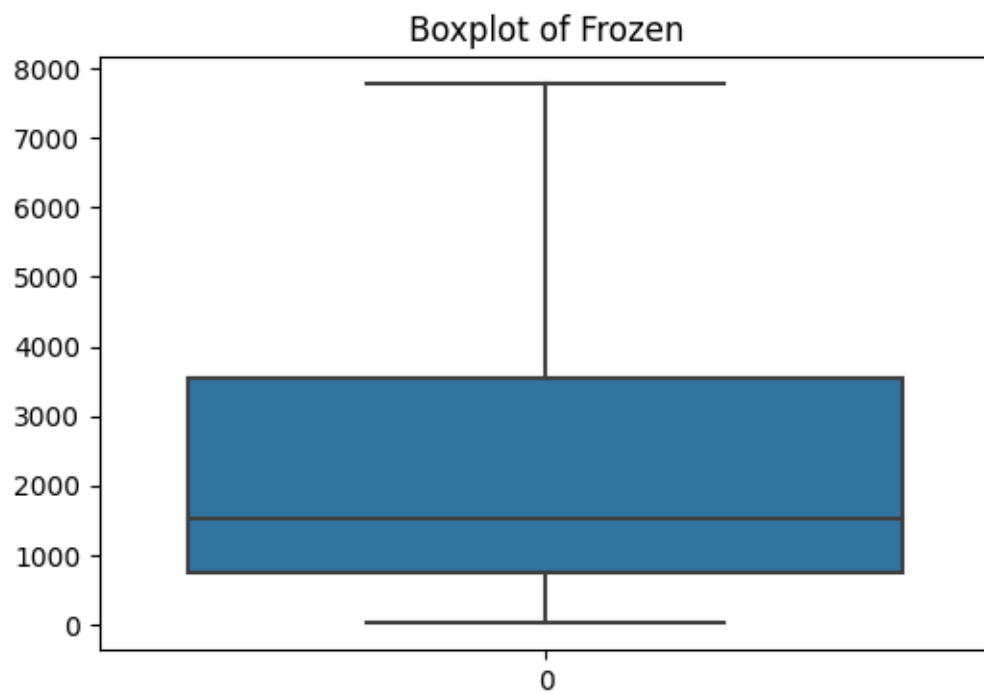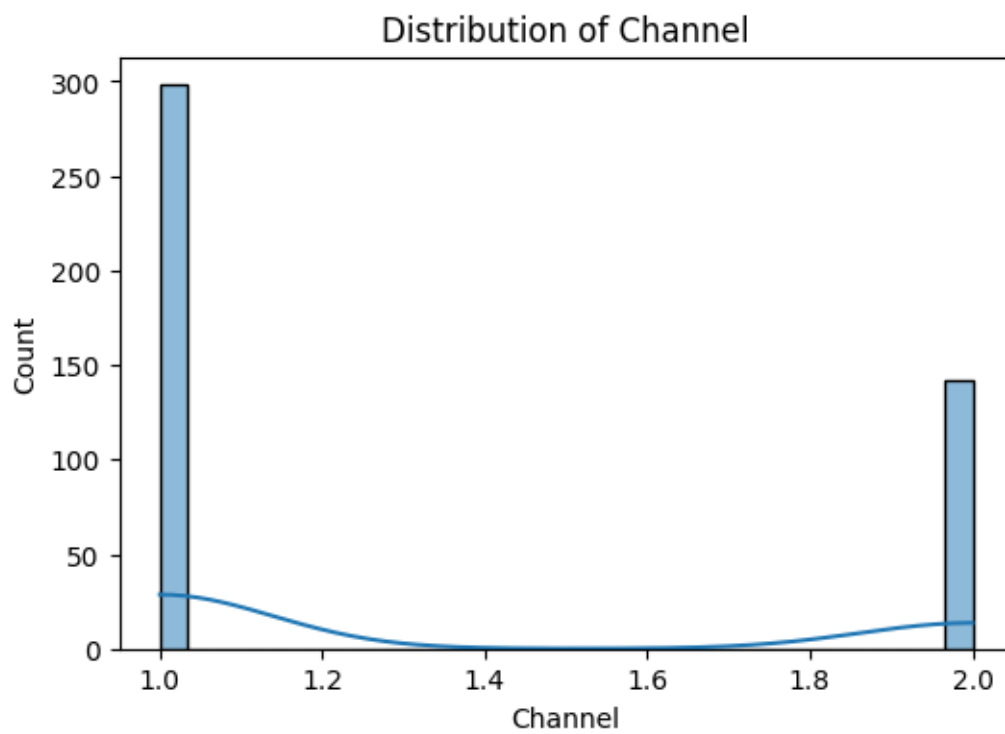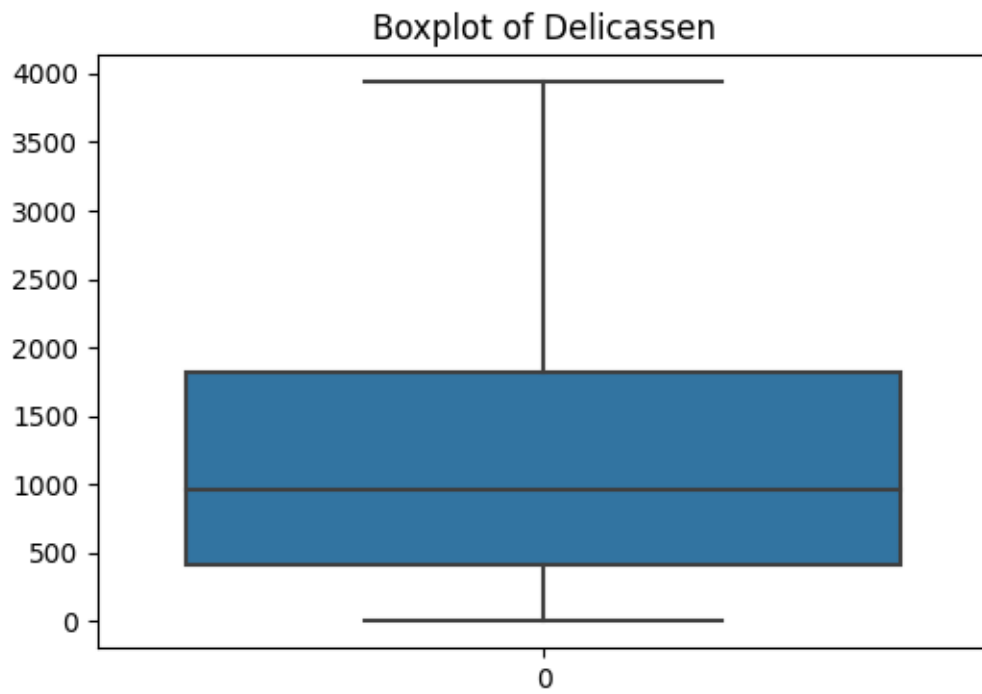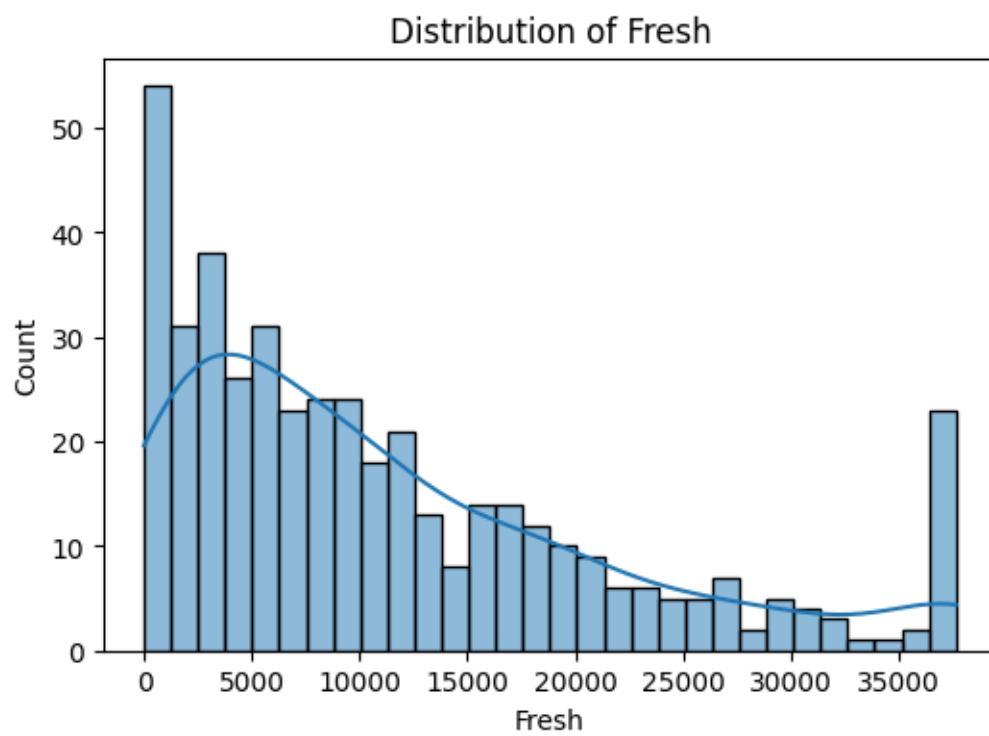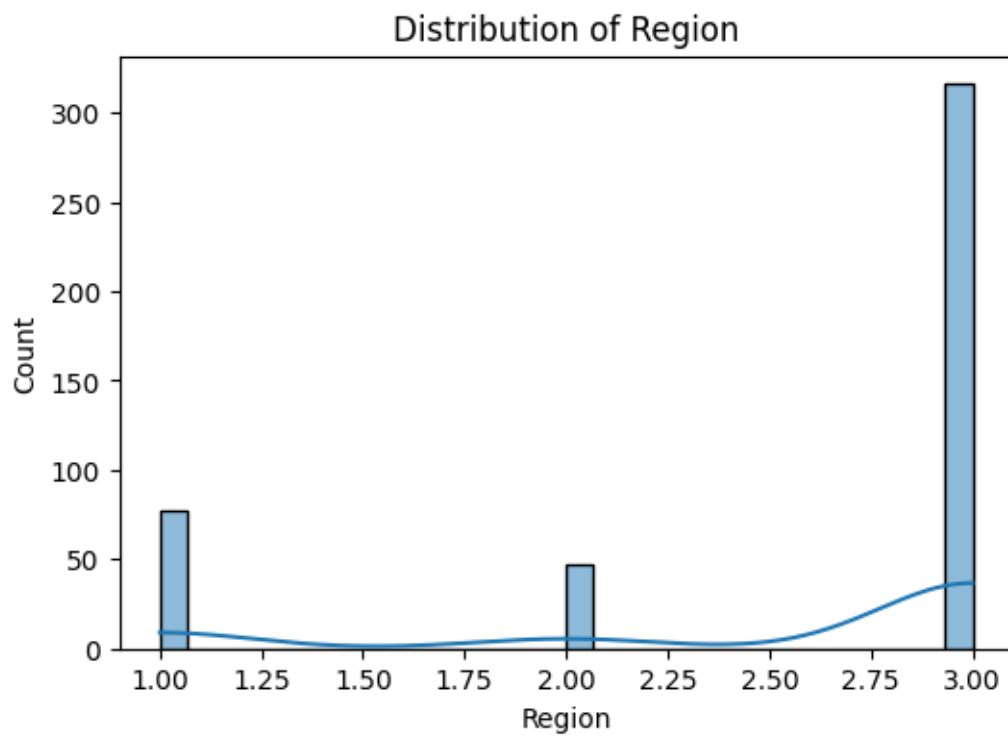
```
Descriptive Statistics:
            Channel      Region          Fresh          Milk       Grocery  \
count    440.000000  440.000000     440.000000    440.000000    440.00000
mean       1.322727    2.543182   11357.568182   5048.592045   7236.37500
std        0.468052    0.774272   10211.542235   4386.377073   6596.53308
min        1.000000    1.000000       3.000000     55.000000      3.00000
25%        1.000000    2.000000    3127.750000   1533.000000   2153.00000
50%        1.000000    3.000000    8504.000000   3627.000000   4755.50000
75%        2.000000    3.000000   16933.750000   7190.250000  10655.75000
max        2.000000    3.000000   37642.750000  15676.125000  23409.87500


             Frozen  Detergents_Paper   Delicassen
count    440.000000        440.000000   440.000000
mean    2507.085795       2392.616477  1266.715341
std     2408.297738       2940.794090  1083.069792
min       25.000000          3.000000     3.000000
25%      742.250000        256.750000   408.250000
50%     1526.000000        816.500000   965.500000
75%     3554.250000       3922.000000  1820.250000
max     7772.250000       9419.875000  3938.250000
Number of duplicate rows:  0
```
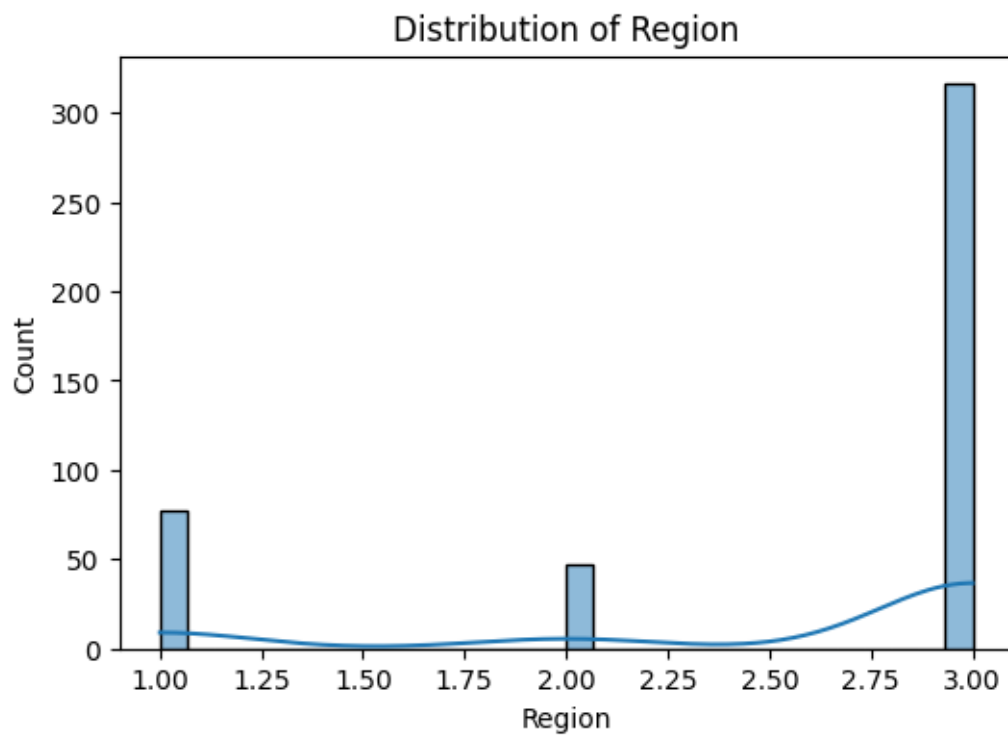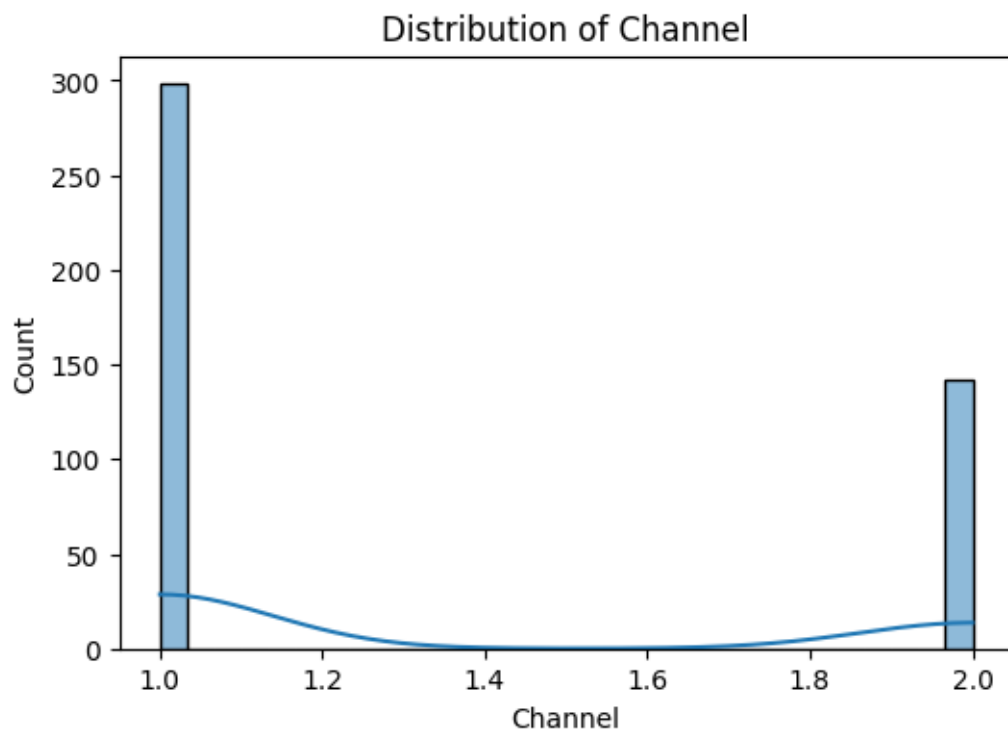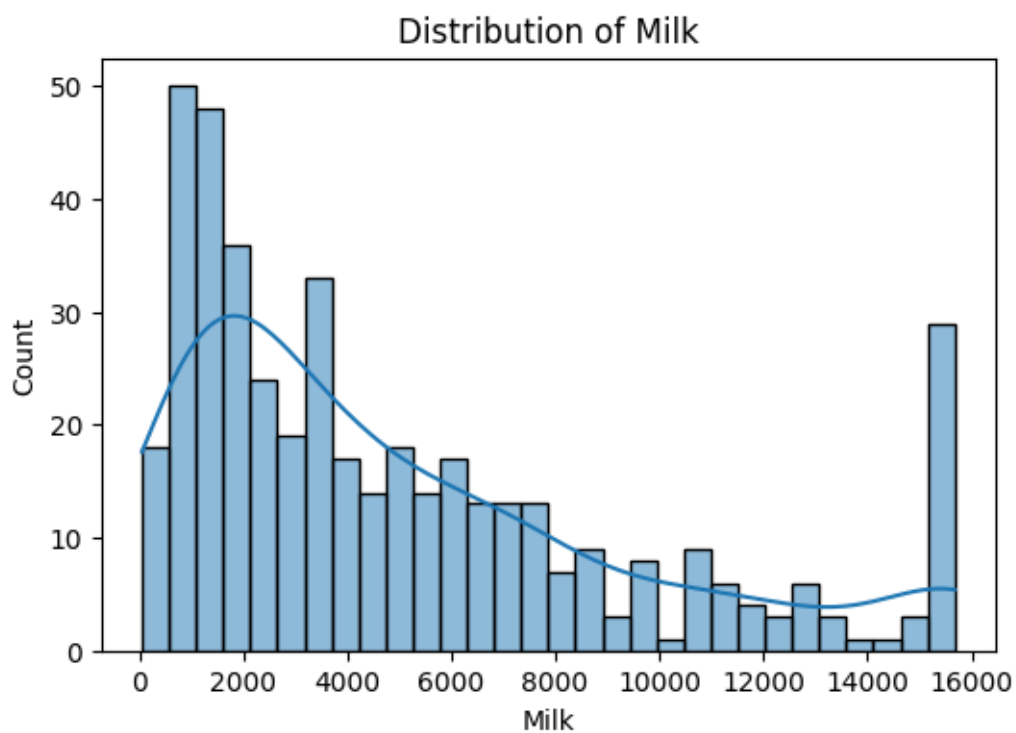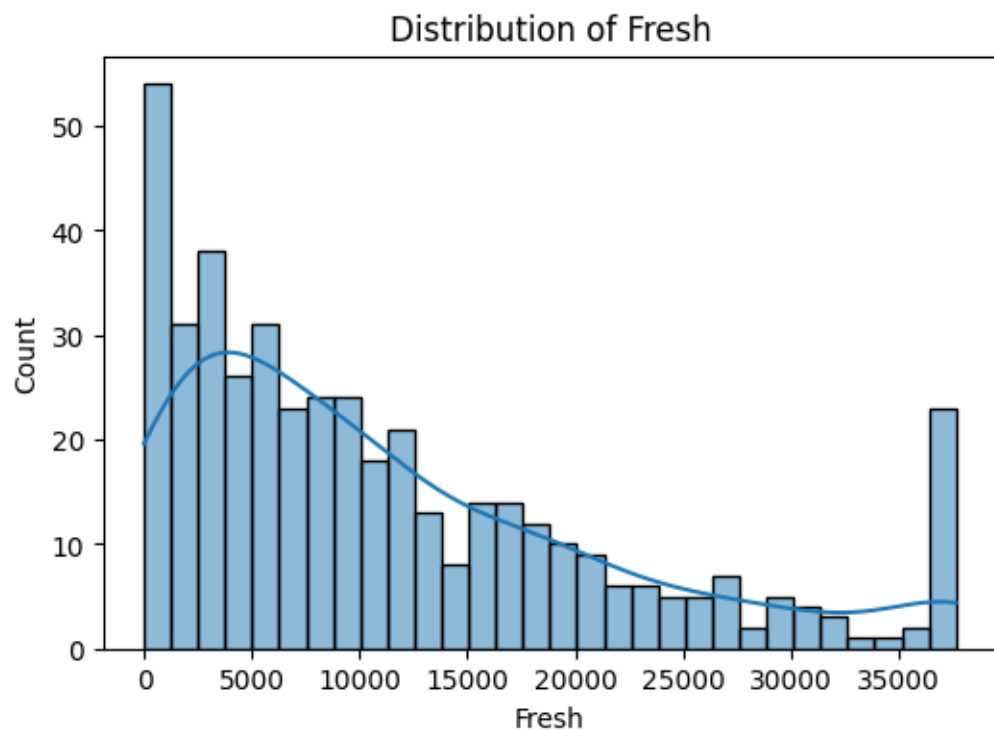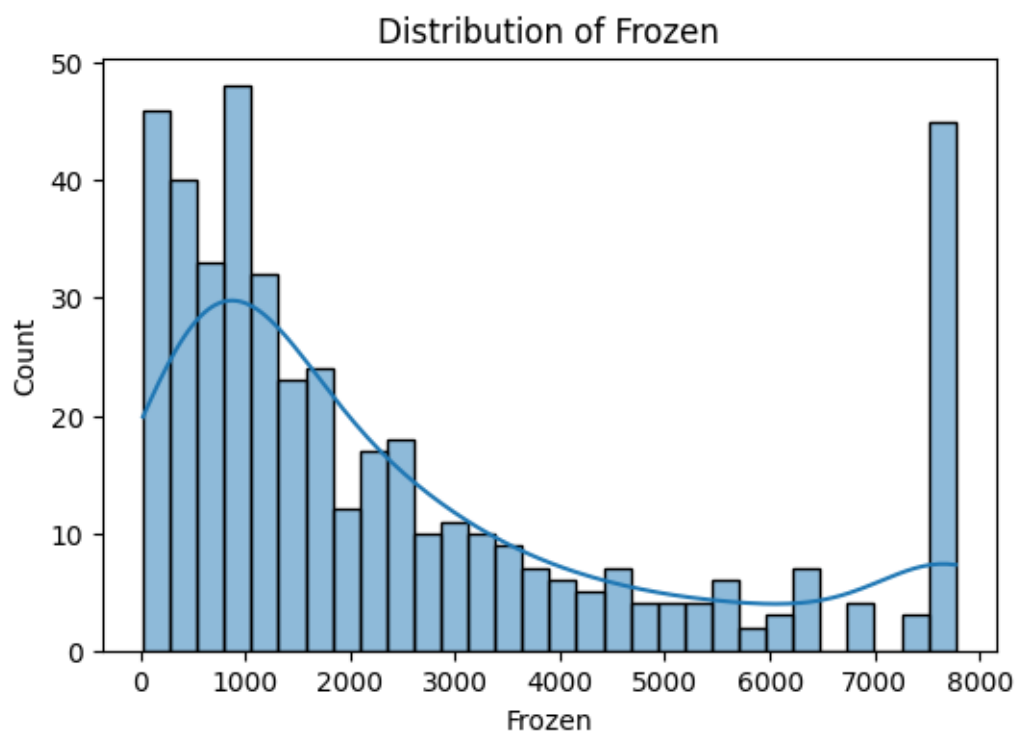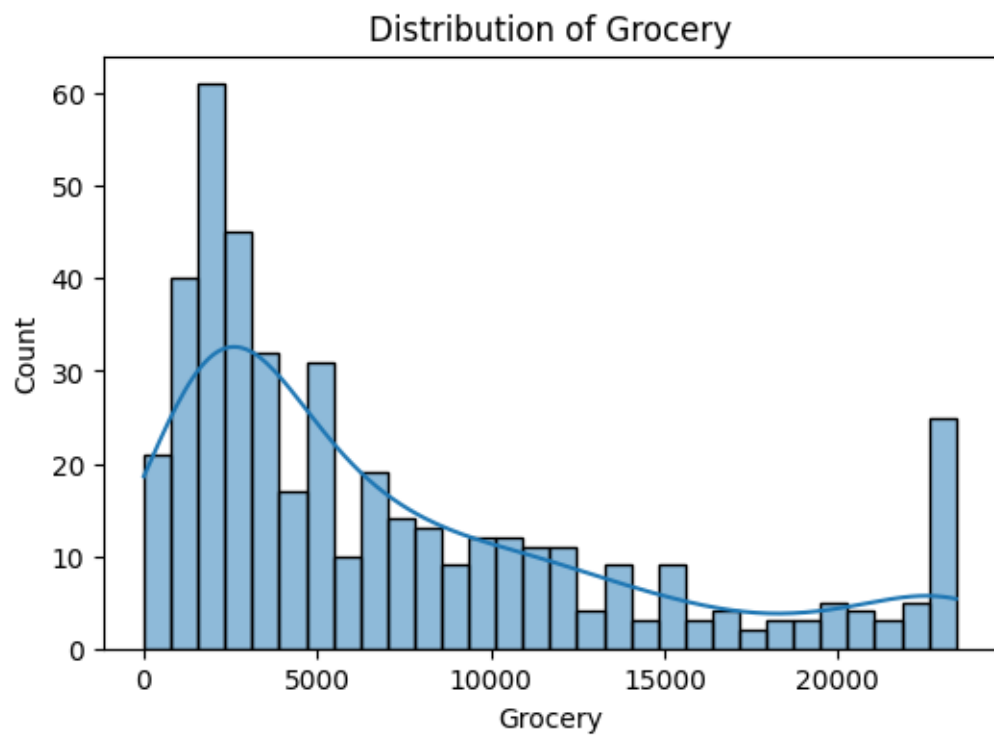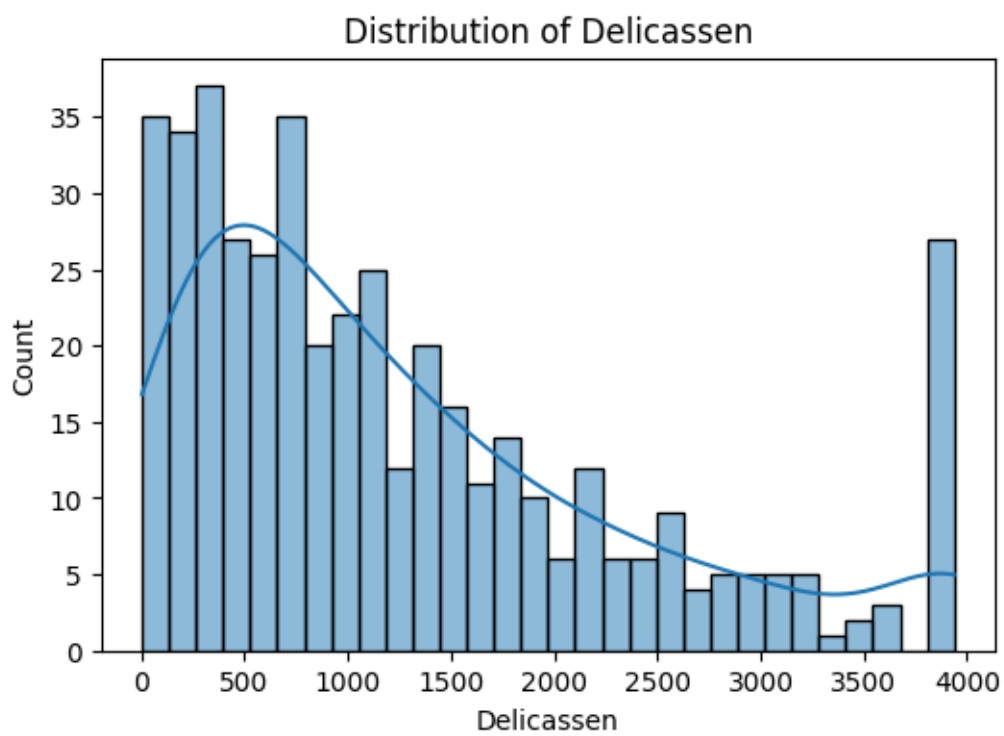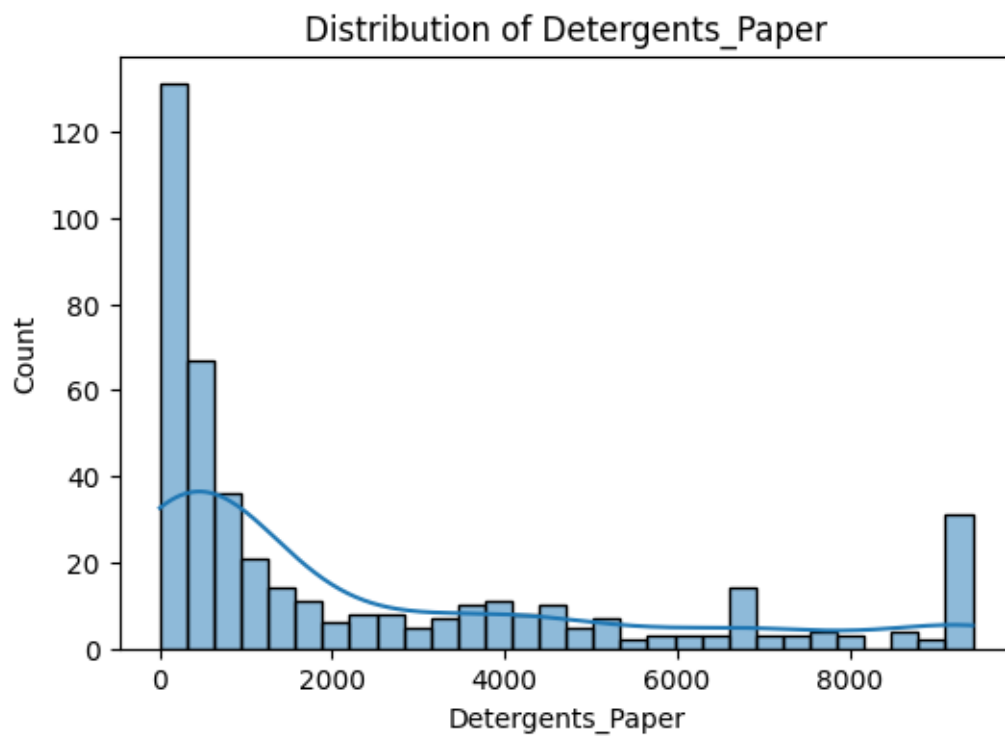
Distribution of Channel

Distribution of Region

Distribution of Fresh

Distribution of Milk

Distribution of Grocery

Distribution of Frozen

Distribution of Detergents_Paper

Distribution of Delicassen

```python
# Heatmap for correlation between variables
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap')
plt.show()
```


Correlation Heatmap
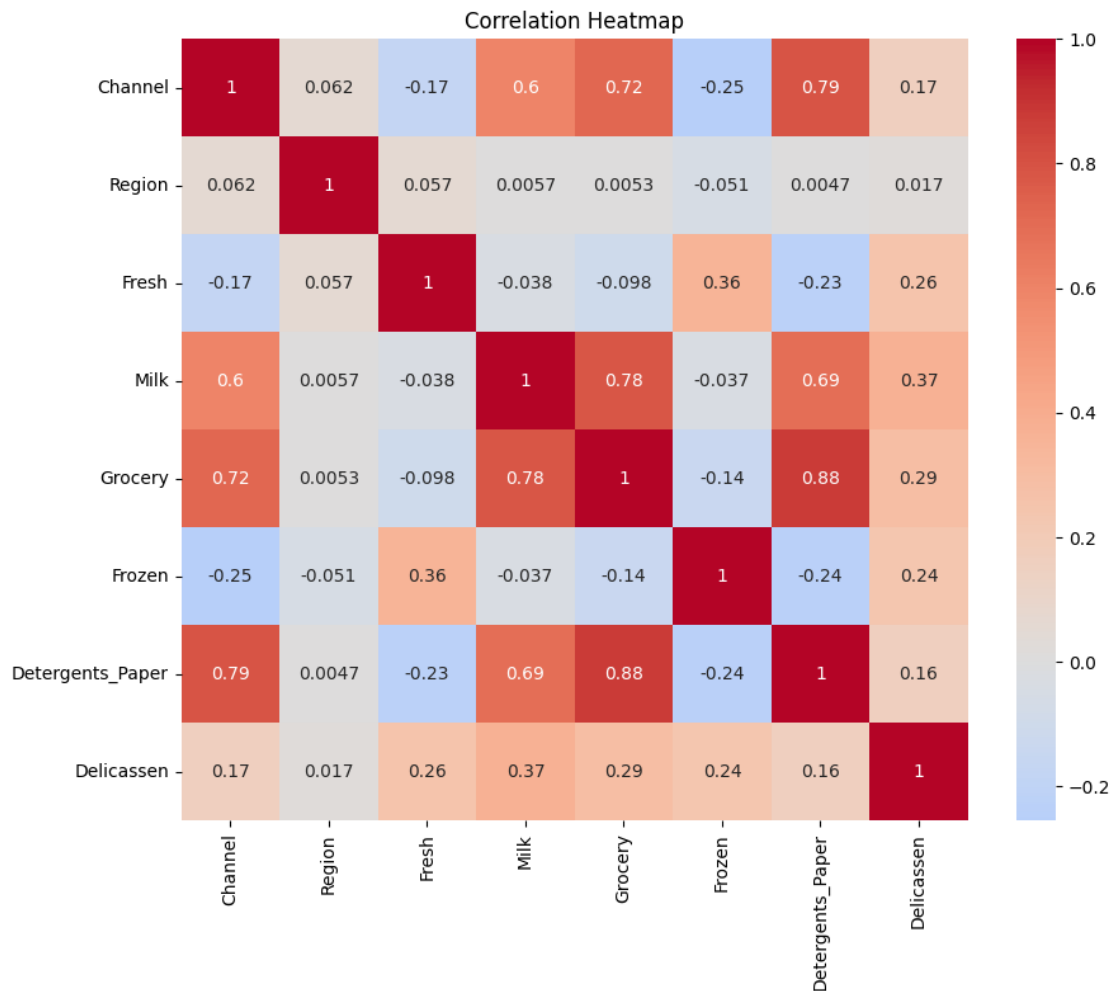
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Calculate WCSS for different number of clusters
wcss = []
```

```python
max_clusters = 15
for i in range(1, max_clusters+1):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)

# Plot the WCSS values
plt.plot(range(1, max_clusters+1), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```

/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
/usr/local/lib/python3.10/dist-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
  warnings.warn(
```

The Elbow Method

```python
from sklearn.cluster import KMeans

# Build the model
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)
kmeans.fit(df)

# Get cluster labels
cluster_labels = kmeans.labels_

# Add cluster labels to your original dataframe
df['Cluster'] = cluster_labels

print(df.head())
```

```
/usr/local/lib/python3.10/dist-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
  warnings.warn(
   Channel  Region     Fresh     Milk  Grocery  Frozen  Detergents_Paper  \
0        2       3   12669.0   9656.0   7561.0   214.0            2674.0
```
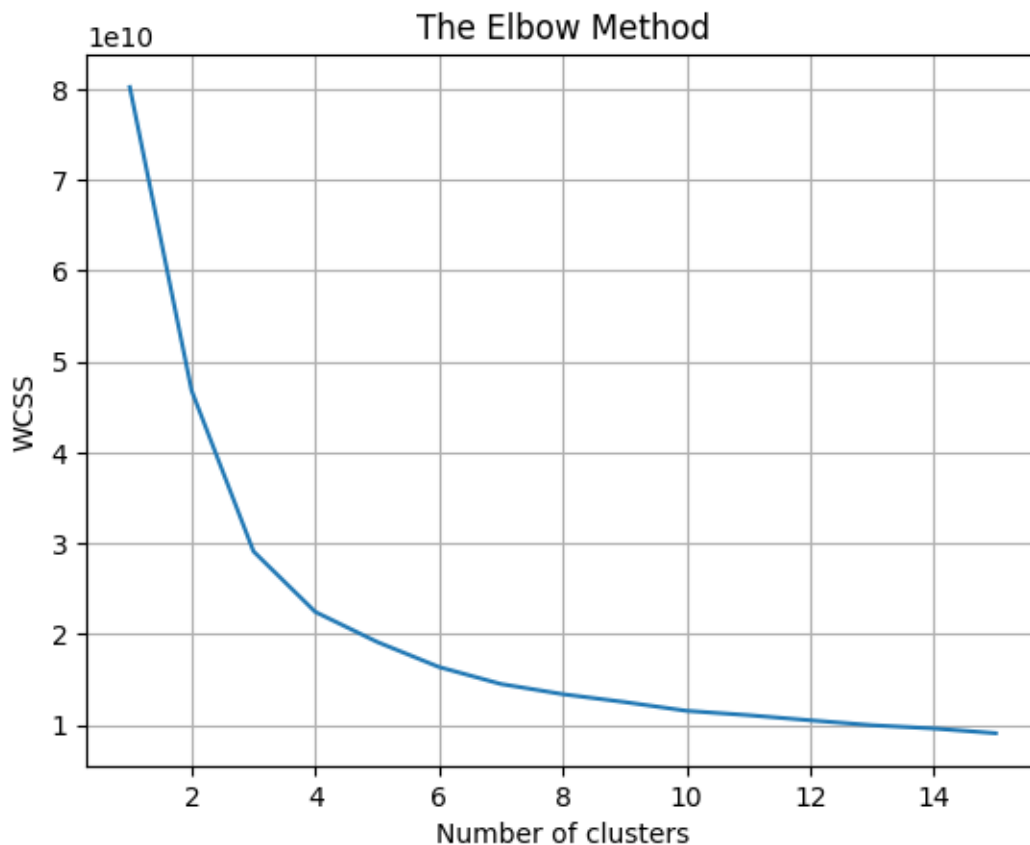
```
1         2         3    7057.0   9810.0   9568.0   1762.0                 3293.0
2         2         3    6353.0   8808.0   7684.0   2405.0                 3516.0
3         1         3   13265.0   1196.0   4221.0   6404.0                  507.0
4         2         3   22615.0   5410.0   7198.0   3915.0                 1777.0

     Delicassen   Cluster
0       1338.00        0
1       1776.00        1
2       3938.25        3
3       1788.00        0
4       3938.25        0
```

```python
# Add cluster labels to the DataFrame
df['Cluster'] = kmeans.labels_

# Check the size of each cluster
print("Cluster Sizes:\n", df['Cluster'].value_counts())

# Check the characteristics of each cluster
for i in range(4):
    print("\nCluster ", i)
    print(df[df['Cluster'] == i].describe())
```

```
Cluster Sizes:
 3    176
0    112
1     94
2     58
Name: Cluster, dtype: int64


Cluster  0
           Channel      Region          Fresh          Milk        Grocery  \
count   112.000000  112.000000     112.000000    112.000000     112.000000
mean      1.214286    2.535714   16051.205357   3135.813616    4211.589286
std       0.412170    0.781873    3763.633078   2524.464860    3150.441587
min       1.000000    1.000000   10379.000000    134.000000       3.000000
25%       1.000000    2.000000   12419.750000   1283.500000    1970.500000
50%       1.000000    3.000000   16195.000000   2252.000000    3203.000000
75%       1.000000    3.000000   18830.250000   4537.000000    5700.250000
max       2.000000    3.000000   24929.000000  15676.125000   14982.000000


              Frozen   Detergents_Paper   Delicassen   Cluster
count     112.000000         112.000000   112.000000     112.0
mean     2988.859375         994.785714  1229.573661       0.0
std      2531.352938        1245.589613   963.527882       0.0
min       118.000000           3.000000    51.000000       0.0
25%      1018.750000         188.500000   514.250000       0.0
50%      2157.500000         456.500000   879.000000       0.0
```

```
75%      4276.000000          1404.000000  1804.500000        0.0
max      7772.250000          6707.000000  3938.250000        0.0


Cluster   1
         Channel     Region        Fresh         Milk       Grocery  \
count  94.000000  94.000000    94.000000    94.000000     94.000000
mean    1.893617   2.489362  5331.893617  10454.450798  17196.140957
std     0.309980   0.799794  5111.448153   3937.245330   4905.345002
min     1.000000   1.000000    18.000000   1266.000000   8852.000000
25%     2.000000   2.000000  1409.500000   7576.000000  12563.250000
50%     2.000000   3.000000  4047.000000  10601.000000  16596.000000
75%     2.000000   3.000000  7870.500000  14316.500000  22288.500000
max     2.000000   3.000000 22925.000000  15676.125000  23409.875000


             Frozen  Detergents_Paper   Delicassen  Cluster
count     94.000000         94.000000    94.000000     94.0
mean    1496.428191       6936.898936  1547.364362      1.0
std     1538.882840       2383.035957  1176.131062      0.0
min       25.000000        241.000000     3.000000      1.0
25%      438.500000       5274.250000   680.000000      1.0
50%      973.000000       6931.500000  1366.500000      1.0
75%     1900.000000       9419.875000  2157.750000      1.0
max     7772.250000       9419.875000  3938.250000      1.0


Cluster   2
         Channel     Region         Fresh         Milk       Grocery  \
count  58.000000  58.000000     58.000000    58.000000     58.000000
mean    1.172414   2.655172  32136.810345   5973.515086   7309.012931
std     0.381039   0.714554   5122.024937   4808.223223   5915.174661
min     1.000000   1.000000  22647.000000    286.000000    471.000000
25%     1.000000   3.000000  27207.500000   2393.000000   2726.250000
50%     1.000000   3.000000  31664.000000   4347.000000   5259.500000
75%     1.000000   3.000000  37642.750000   7829.500000   9344.000000
max     2.000000   3.000000  37642.750000  15676.125000  23409.875000


             Frozen  Detergents_Paper   Delicassen  Cluster
count     58.000000         58.000000    58.000000     58.0
mean    4170.017241       1417.426724  1967.702586      2.0
std     2841.060439       2055.702539  1267.507352      0.0
min      127.000000         10.000000     3.000000      2.0
25%     1370.750000        250.250000  1037.250000      2.0
50%     3662.000000        617.500000  1821.500000      2.0
75%     7772.250000       1428.000000  2910.250000      2.0
max     7772.250000       9419.875000  3938.250000      2.0


Cluster   3
         Channel      Region         Fresh         Milk       Grocery  \
count  176.000000  176.000000    176.000000   176.000000    176.000000
```

```
mean      1.136364      2.539773      4741.261364      3073.790483      3817.880682
std       0.344153      0.777254      3072.006036      2492.137013      2790.348628
min       1.000000      1.000000         3.000000        55.000000       137.000000
25%       1.000000      2.000000      2116.000000      1109.000000      1739.250000
50%       1.000000      3.000000      4659.500000      2268.000000      2765.500000
75%       1.000000      3.000000      7369.250000      4394.250000      5494.500000
max       2.000000      3.000000     10290.000000     15676.125000     12400.000000

              Frozen   Detergents_Paper   Delicassen   Cluster
count    176.000000         176.000000   176.000000     176.0
mean    2192.274148        1176.454545   909.451705       3.0
std     2210.017535        1473.393792   872.339683       0.0
min       47.000000           5.000000     3.000000       3.0
25%      587.750000         216.500000   308.250000       3.0
50%     1310.000000         472.500000   674.500000       3.0
75%     2964.250000        1545.000000  1154.750000       3.0
max     7772.250000        7271.000000  3938.250000       3.0
```

```python
# Calculate the mean values for each feature per cluster
cluster_means = df.groupby('Cluster').mean()

# Transpose the DataFrame so that the features are the rows (this will make
 ↪plotting easier)
cluster_means = cluster_means.transpose()

# Create bar plot for each feature
for feature in cluster_means.index:
    cluster_means.loc[feature].plot(kind='bar', figsize=(8,6))
    plt.title(feature)
    plt.ylabel('Mean Value')
    plt.xticks(ticks=range(4), labels=['Cluster 0', 'Cluster 1', 'Cluster 2',
 ↪'Cluster 3'])
    plt.show()
```
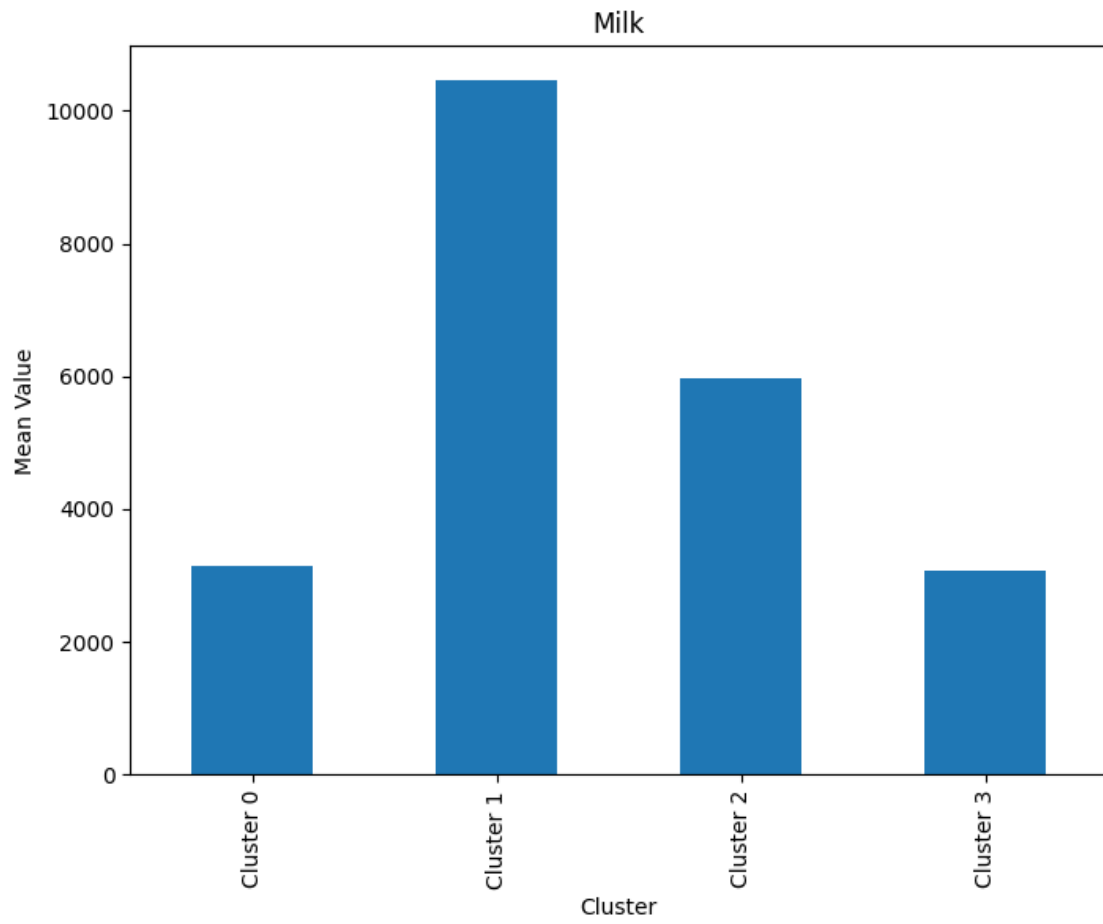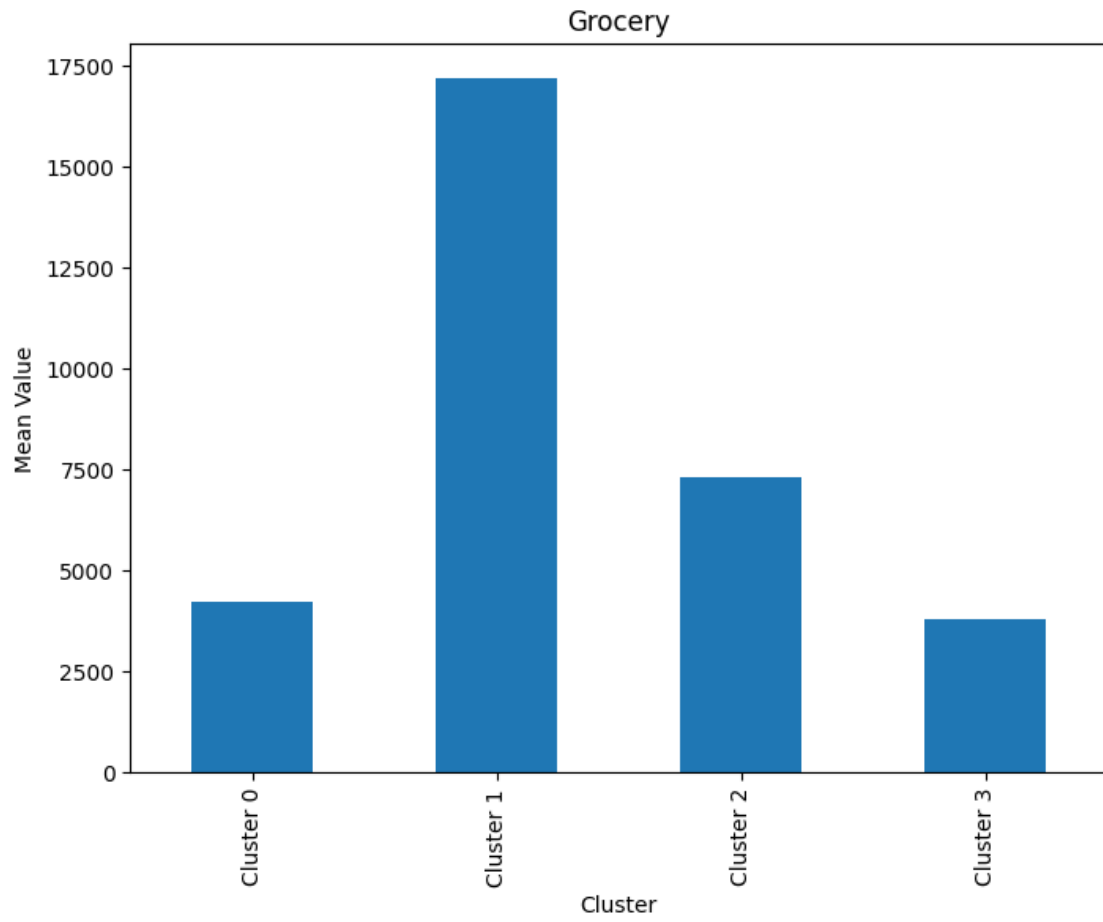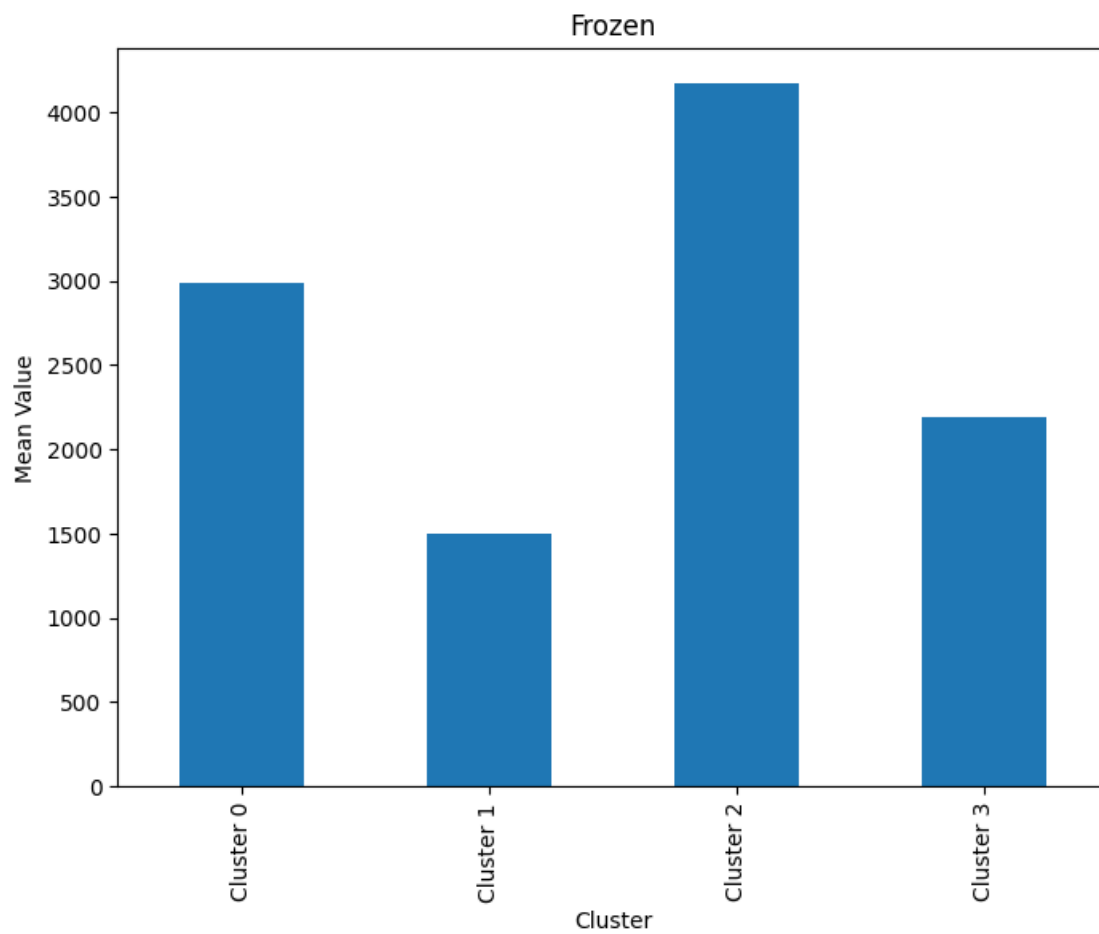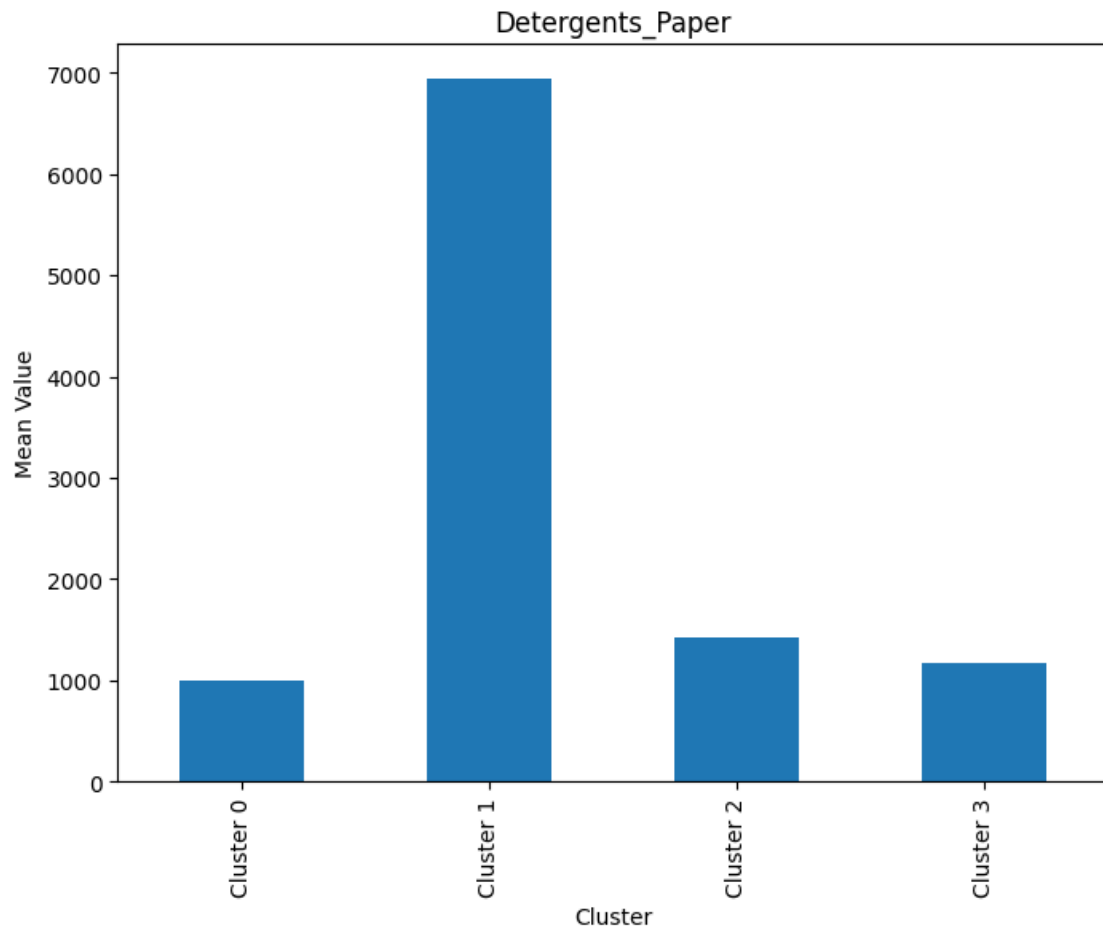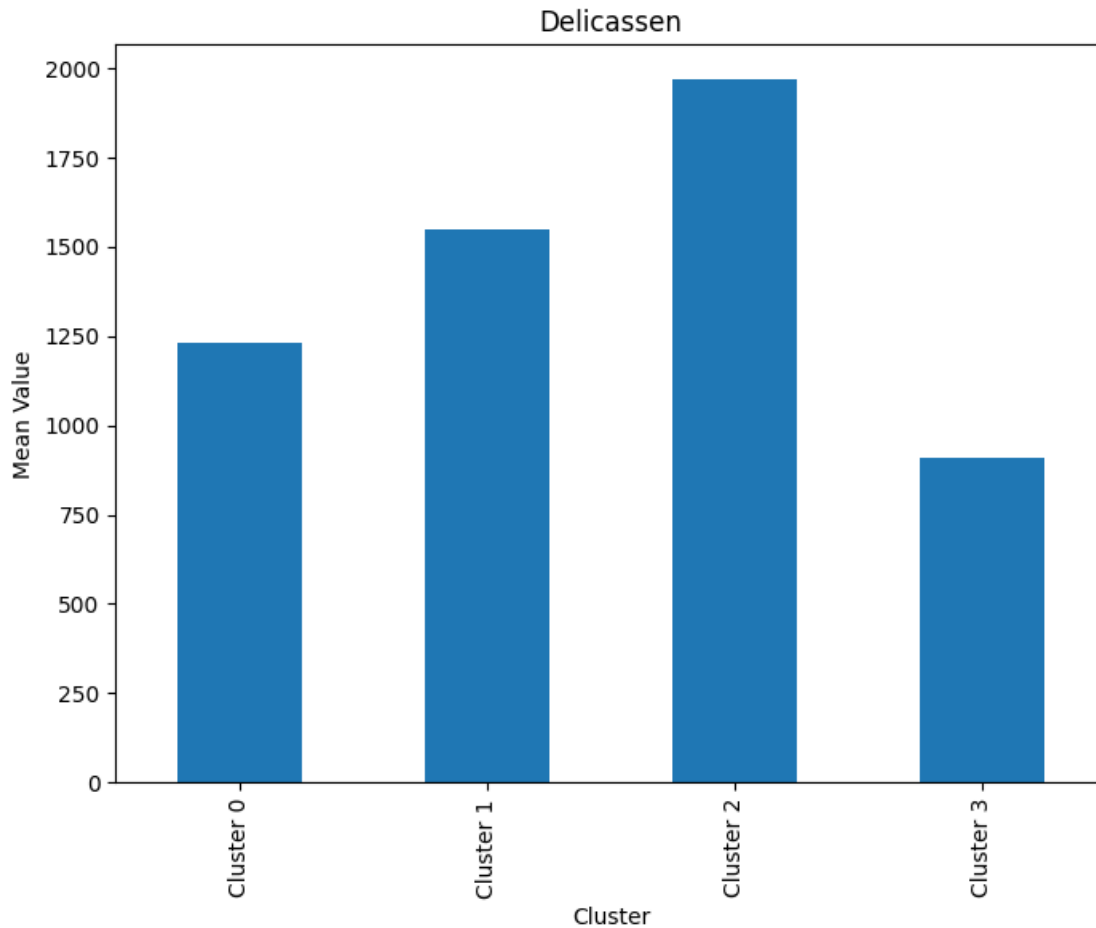
Channel

Region

36

Fresh

Grocery

39

Frozen

Detergents_Paper

Delicassen
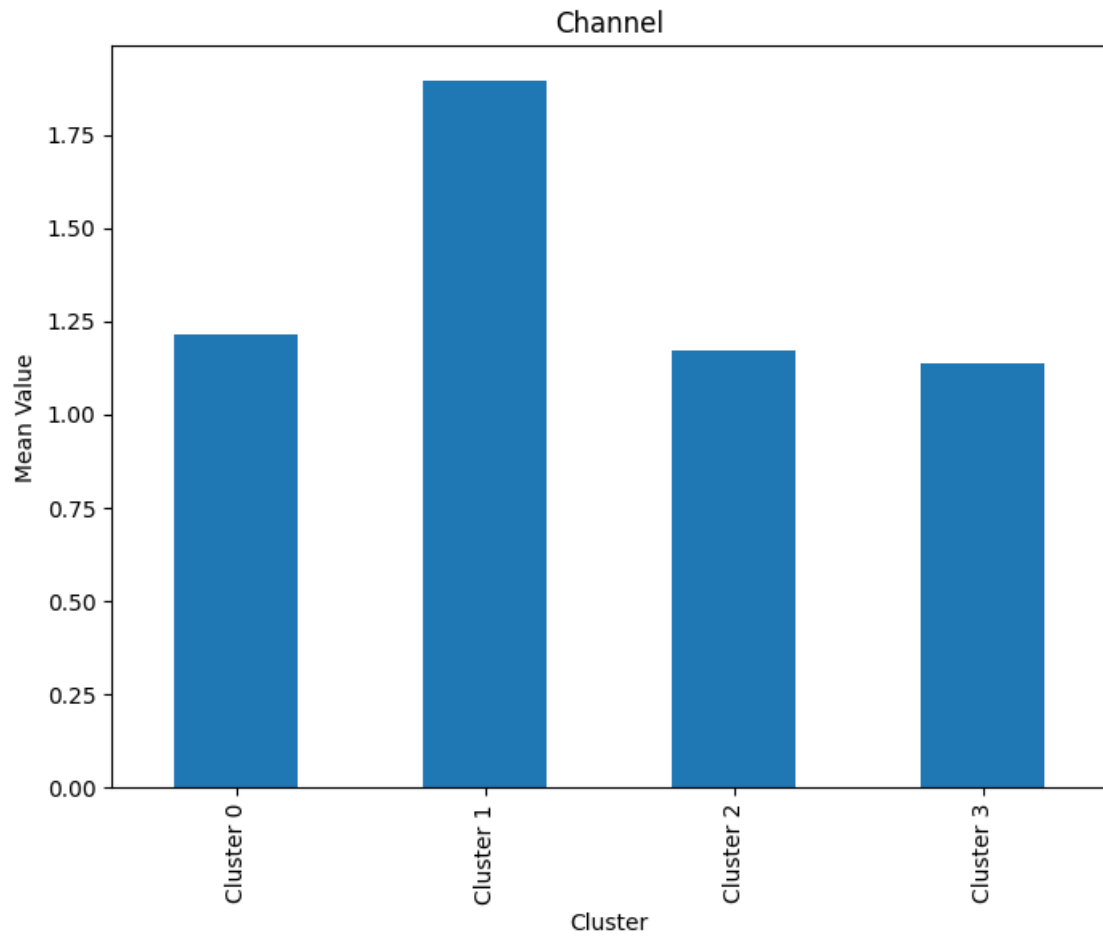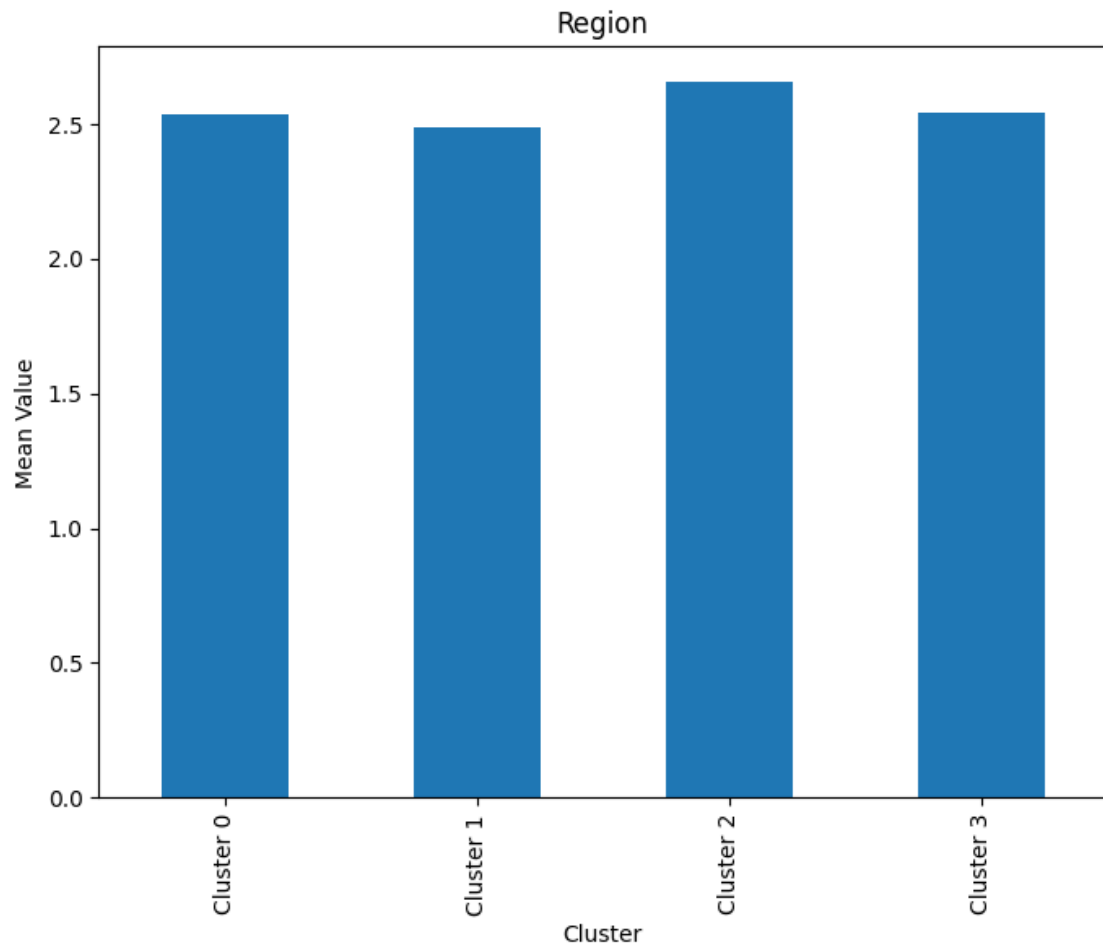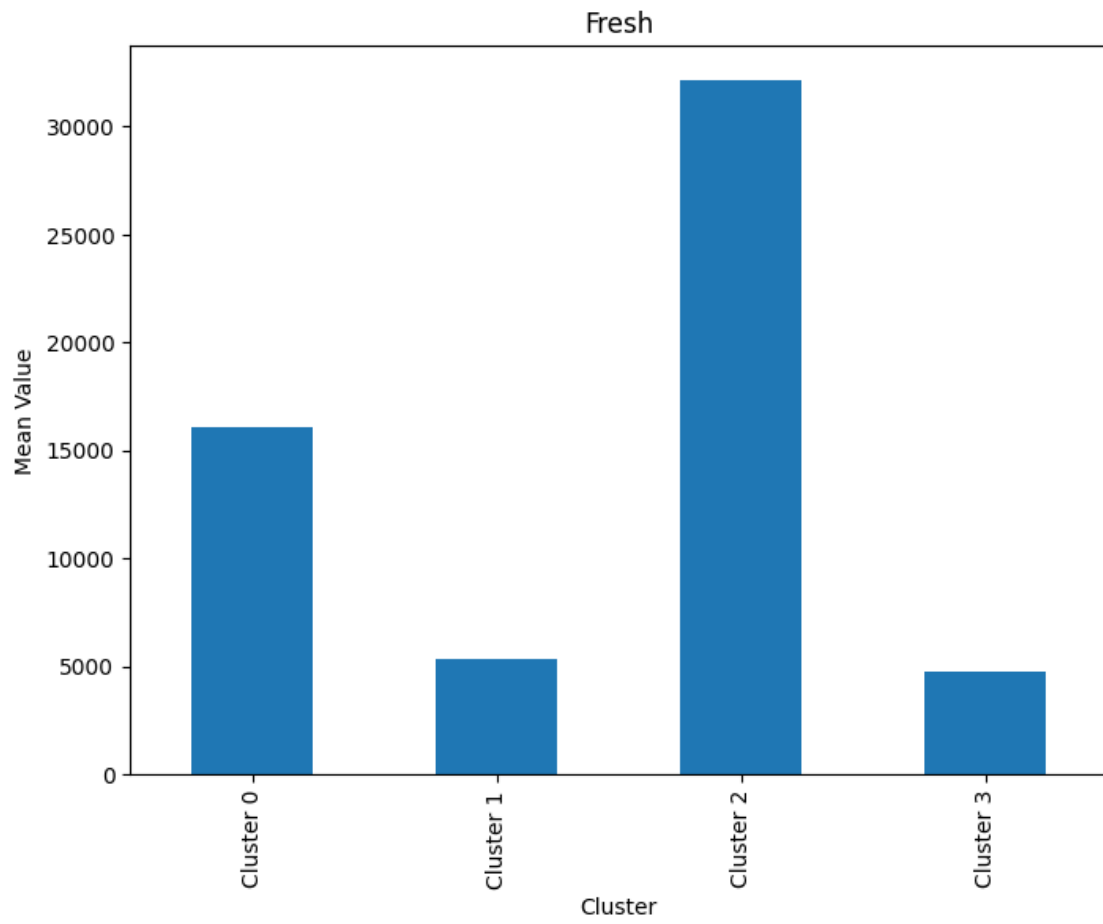
```python
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Apply PCA and fit the features selected
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df.drop('Cluster', axis=1))

# Create a DataFrame with the two components
PCA_components = pd.DataFrame(principalComponents, columns=['Principal␣
 ↪Component 1', 'Principal Component 2'])

# Concatenate the clusters labels to the DataFrame
PCA_components['Cluster'] = df['Cluster']

# Plot the clustered dataset
plt.figure(figsize=(8,6))
```
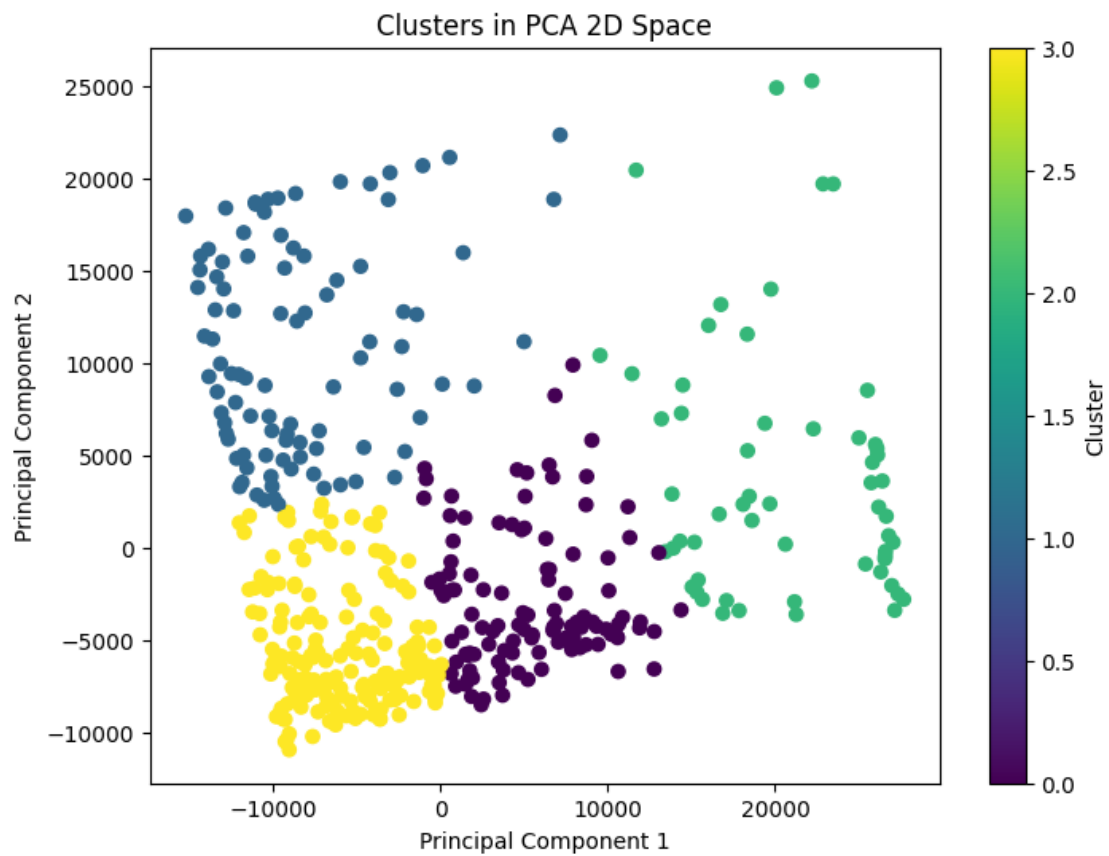
```
plt.scatter(PCA_components['Principal Component 1'], PCA_components['Principal␣
  ↪Component 2'], c=PCA_components['Cluster'])
plt.title('Clusters in PCA 2D Space')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

Conclusion:

Utilizing Clustered Data:

Targeted Marketing: Customize marketing strategies for each cluster.

Inventory Management: Optimize stock levels based on cluster preferences.

Supply Chain Optimization: Tailor delivery schedules to cluster needs.

Product Recommendations: Offer personalized product suggestions.

Customer Service: Adapt service based on cluster preferences.

Effect of Delivery Scheme on Customer Segments:

Cluster 0: Flexible delivery for diverse product needs.

Cluster 1: Subscription-based for essential items.

Cluster 2: Freshness guarantee with quick delivery.

Cluster 3: Cost-effective and efficient delivery options.

Collect feedback from each cluster to refine delivery schemes for better satisfaction and loyalty.