Experiment No. 5
Apply appropriate Unsupervised Learning Technique on the Wholesale Customers Dataset
Date of Performance:
Date of Submission:

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

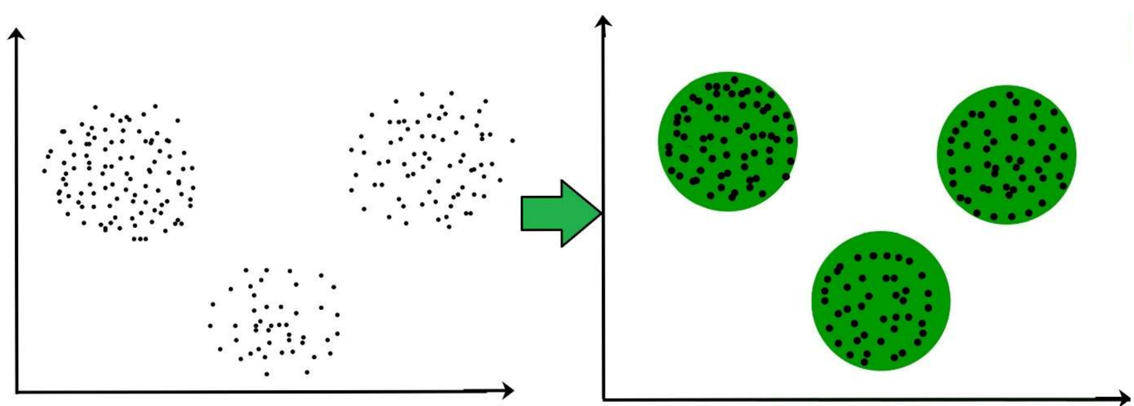
Objective: Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

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)

Conclusion:

1.The customers are divide into 4 clusters based on elbow method as shown in the code.
2.Utilizing clustered data generated through techniques like K-means clustering can be instrumental for various business applications. For instance, it enables the grouping of customers with similar buying patterns, facilitating the creation of more effective marketing strategies. Additionally, it can be employed to offer product recommendations to customers based on their cluster affiliation and to discover associations between frequently co-purchased items within each cluster, among other possibilities.
3.The distinct customer segments within the clusters may respond differently to a specific delivery scheme, as their individual needs and expectations can be quite diverse. To address this variation, an approach that assesses the alignment of a proposed delivery scheme with the unique characteristics and preferences of each customer segment is essential.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv("/content/Wholesale customers data.csv")

df.head()

   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

df.shape

(440, 8)

df.describe()

          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
```

```
df.info()
```

```
<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
```

```
df.dtypes
```

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

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

```
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

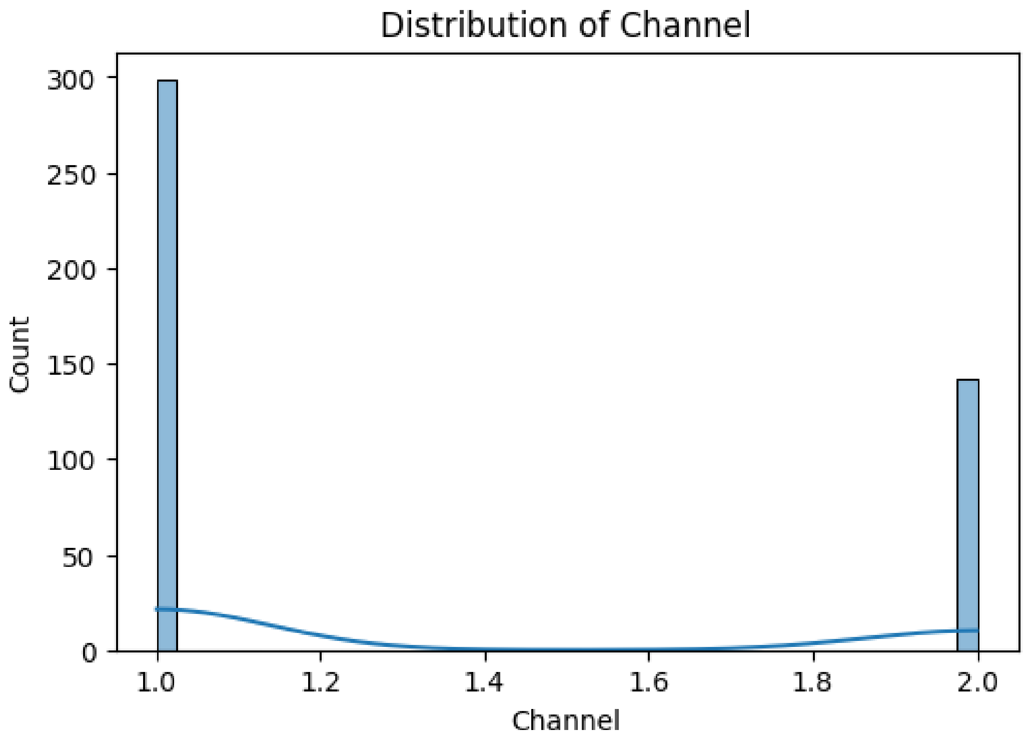
```
df.corr()
```

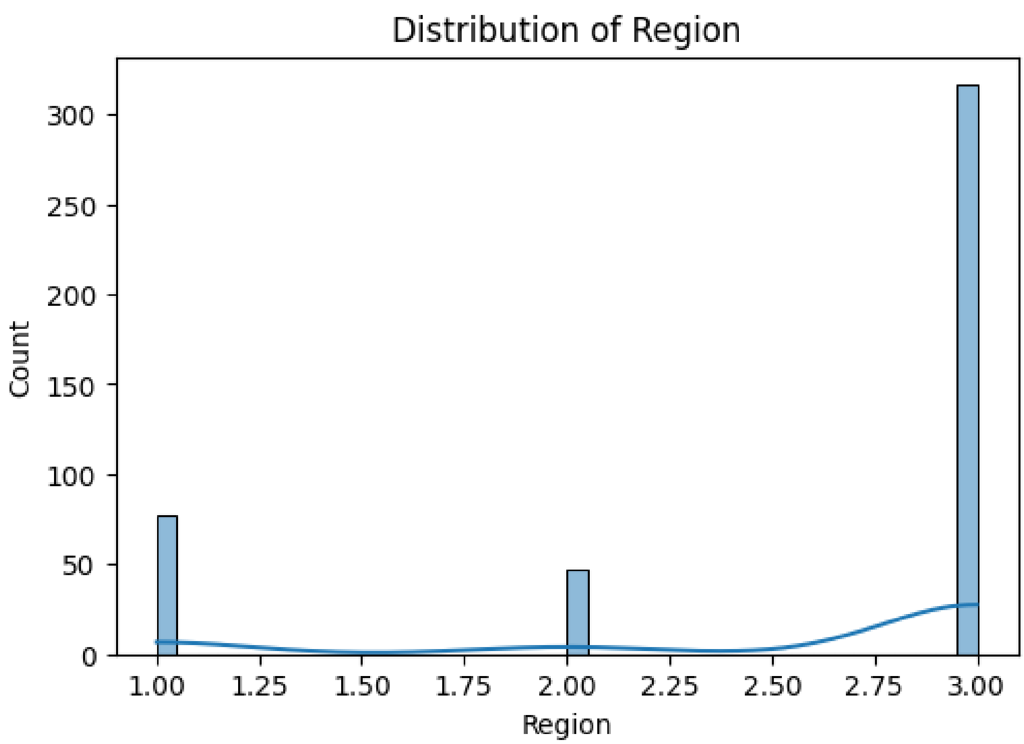
```
                   Channel    Region     Fresh      Milk   Grocery
Frozen  \
Channel           1.000000  0.062028 -0.169172  0.460720  0.608792 -
0.202046
Region            0.062028  1.000000  0.055287  0.032288  0.007696 -
0.021044
Fresh            -0.169172  0.055287  1.000000  0.100510 -0.011854
0.345881
Milk              0.460720  0.032288  0.100510  1.000000  0.728335
0.123994
Grocery           0.608792  0.007696 -0.011854  0.728335  1.000000 -
0.040193
Frozen           -0.202046 -0.021044  0.345881  0.123994 -0.040193
1.000000
Detergents_Paper  0.636026 -0.001483 -0.101953  0.661816  0.924641 -
0.131525
Delicassen        0.056011  0.045212  0.244690  0.406368  0.205497
0.390947

                  Detergents_Paper  Delicassen
Channel                   0.636026    0.056011
Region                   -0.001483    0.045212
Fresh                    -0.101953    0.244690
Milk                      0.661816    0.406368
Grocery                   0.924641    0.205497
Frozen                   -0.131525    0.390947
Detergents_Paper          1.000000    0.069291
Delicassen                0.069291    1.000000
```

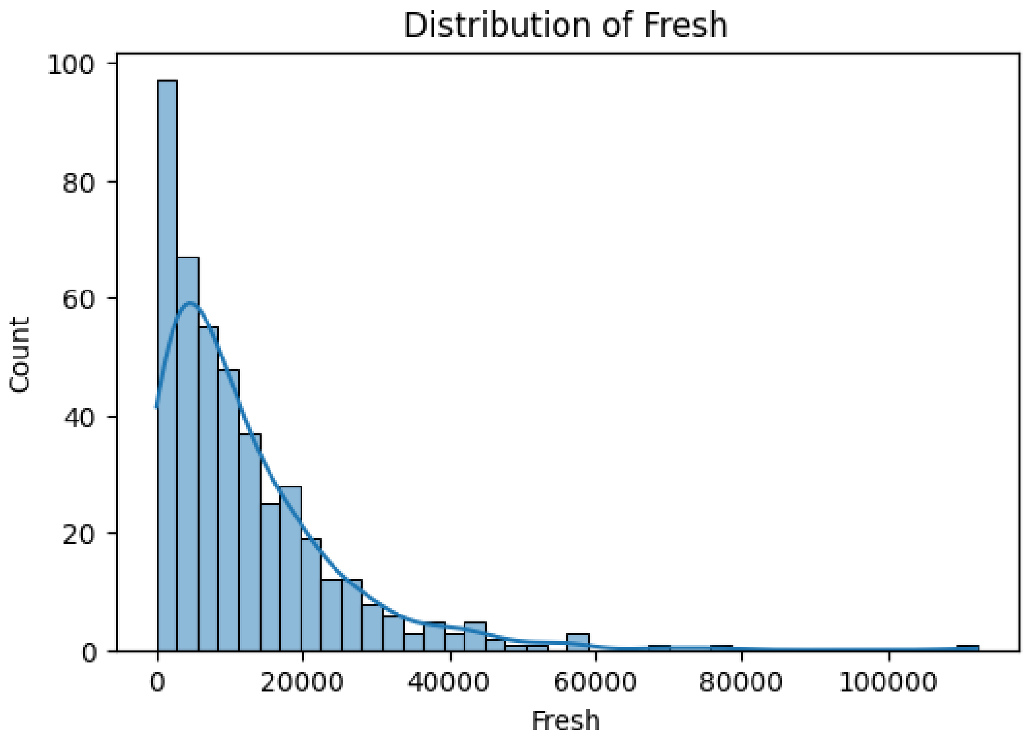
```
df.columns
```

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

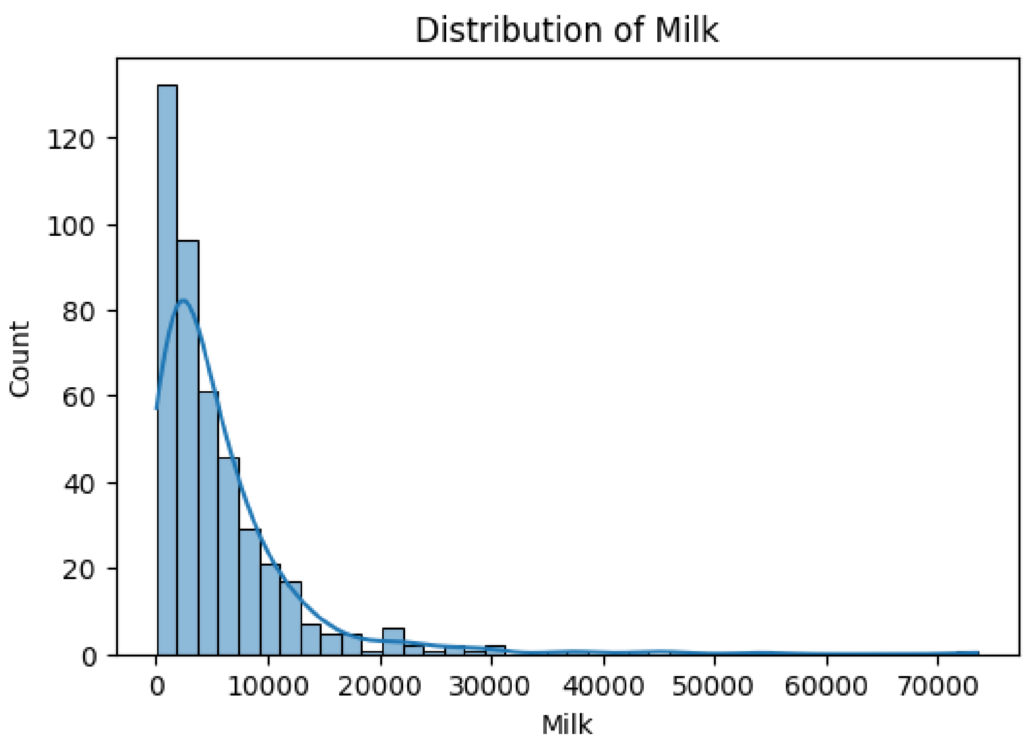
```
## Data distribution:

for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[column], bins=40, kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()
```


Distribution of Channel
Count
300
250
200
150
100
50
0
1.0
1.2
1.4
1.6
1.8
2.0
Channel

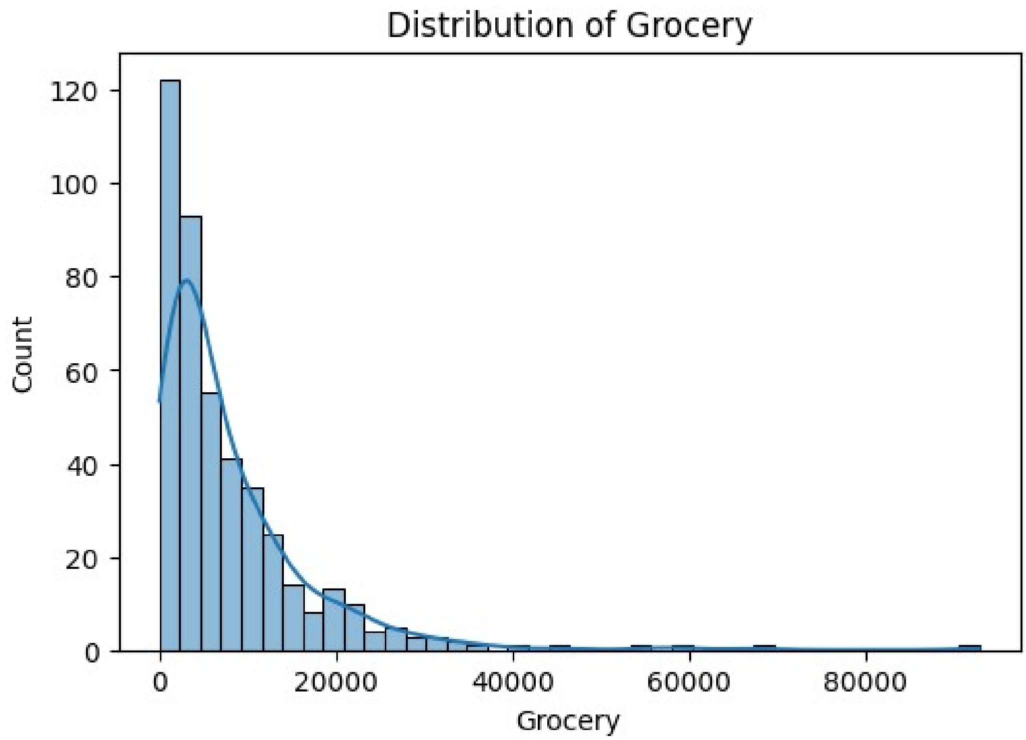

Distribution of Region
Count
300
250
200
150
100
50
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Region

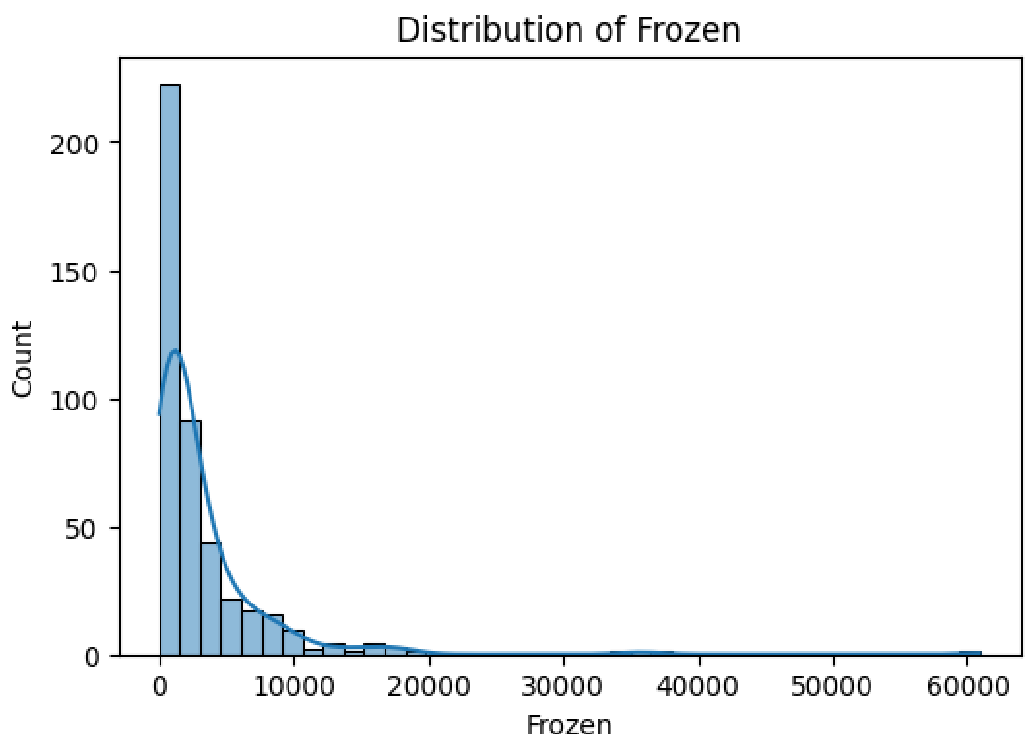


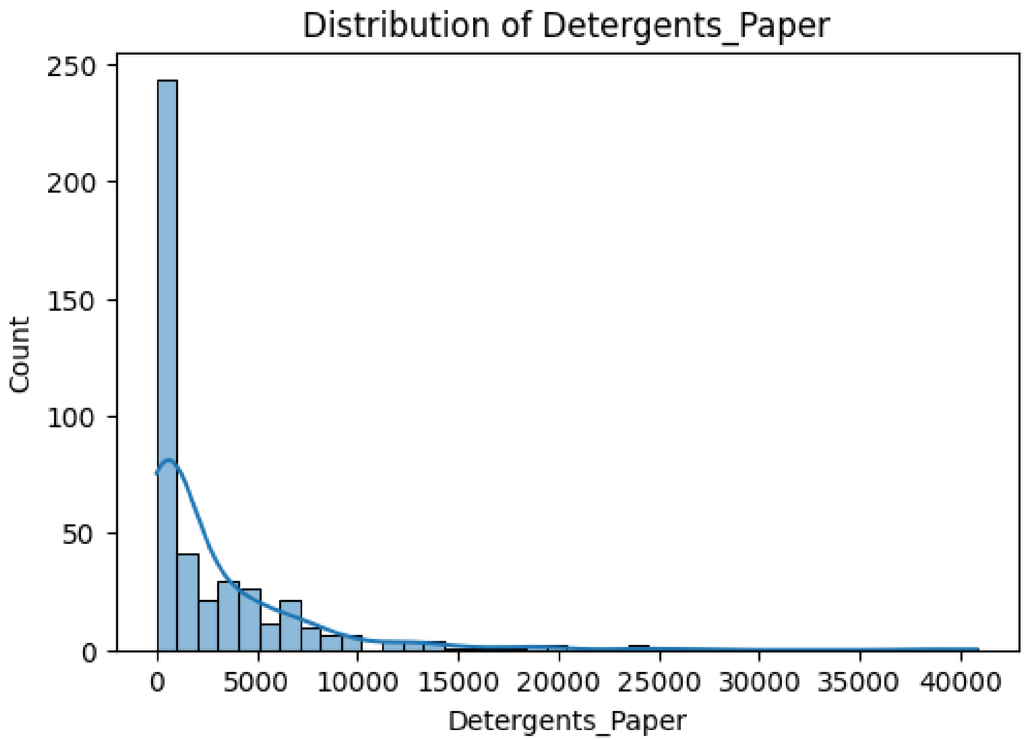
Distribution of Fresh
Count
0
20
40
60
80
100
0
20000
40000
60000
80000
100000
Fresh

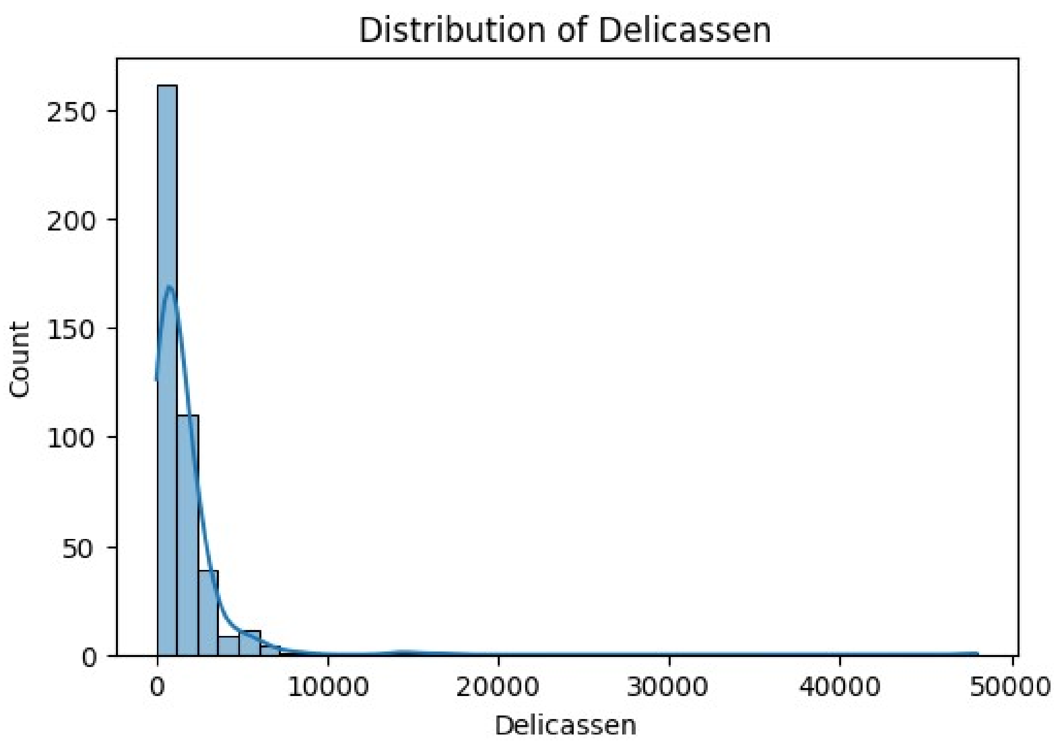


Distribution of Milk
Count
0
20
40
60
80
100
120
0
10000
20000
30000
40000
50000
60000
70000
Milk


Distribution of Grocery
Count
120
100
80
60
40
20
0
0
20000
40000
60000
80000
Grocery

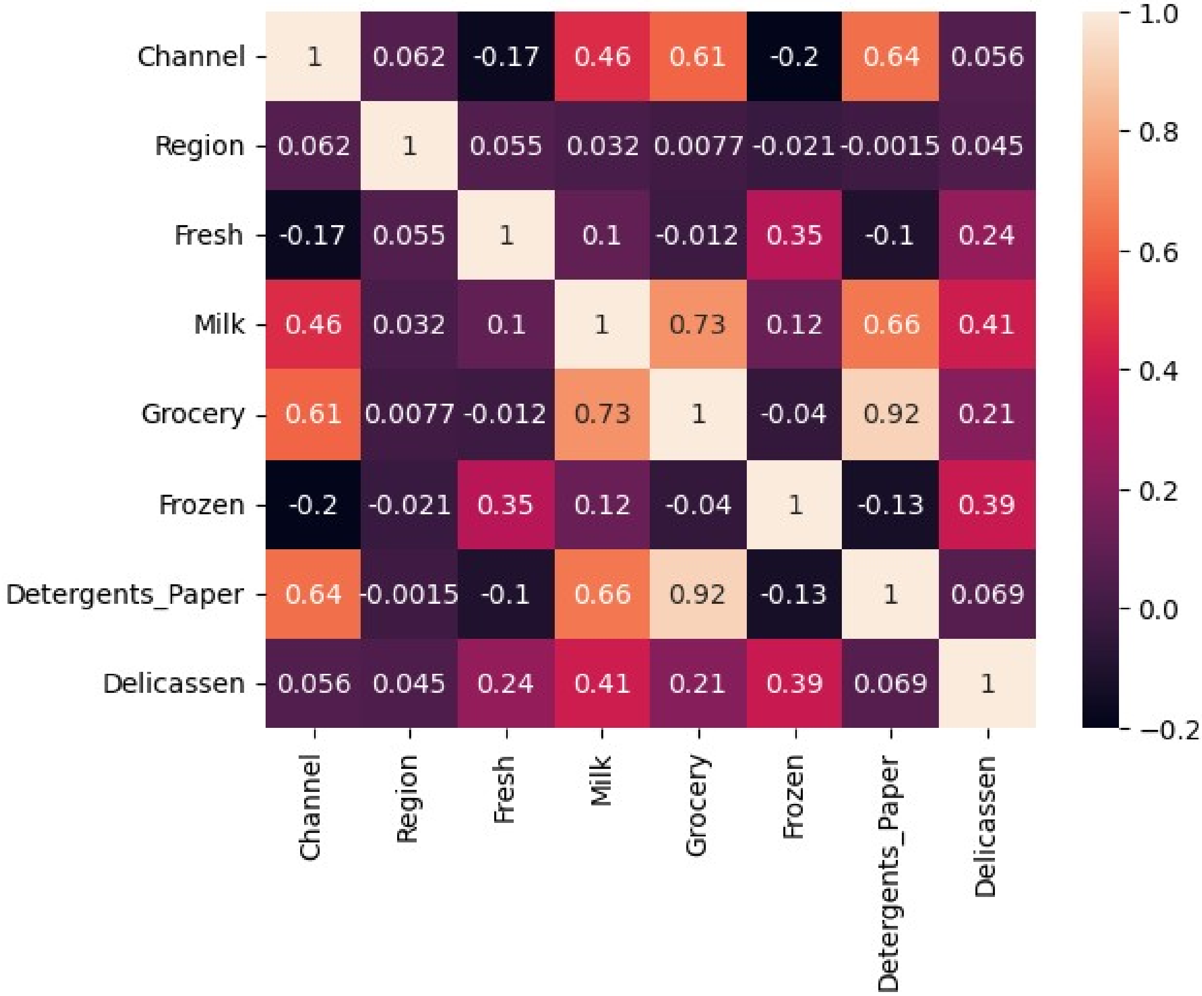

Distribution of Frozen
Count
200
150
100
50
0
0
10000
20000
30000
40000
50000
60000
Frozen



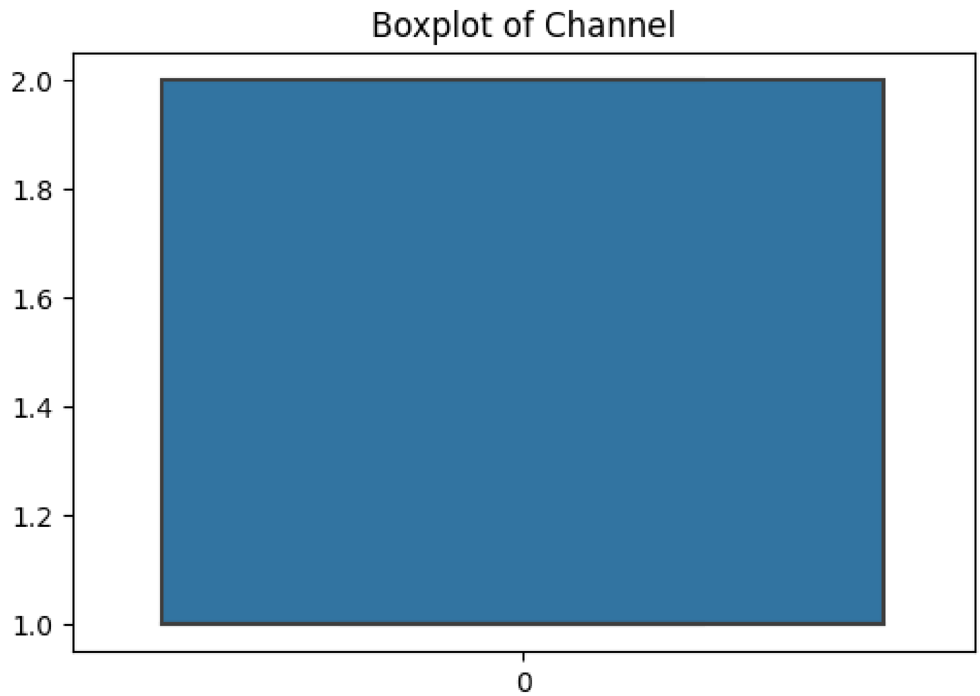


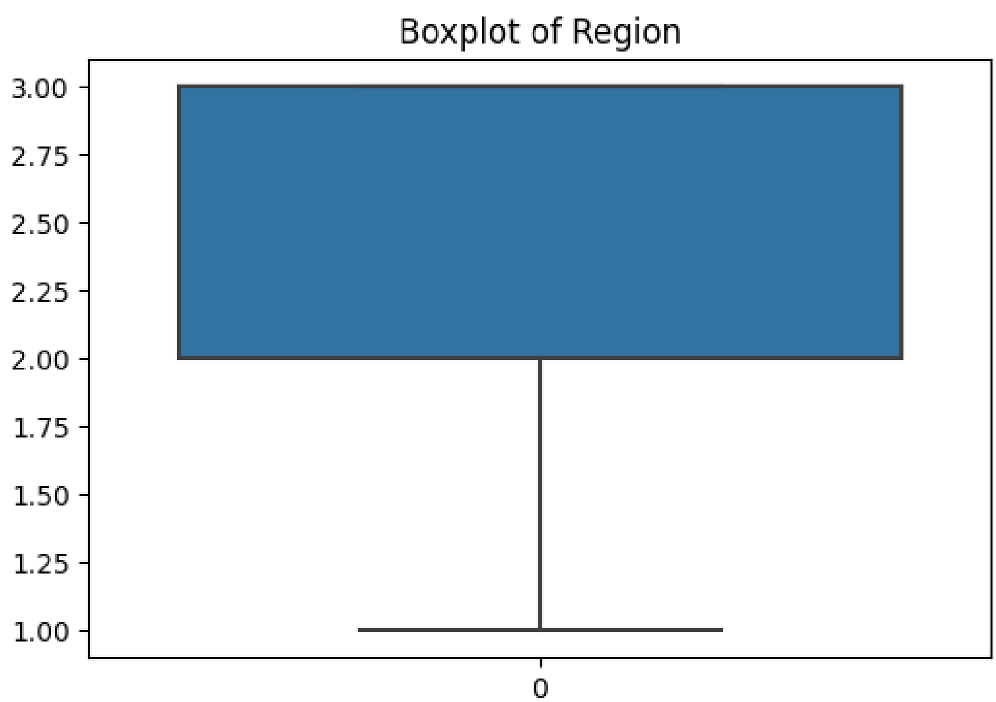

```
sns.heatmap(df.corr() , annot=True)
```

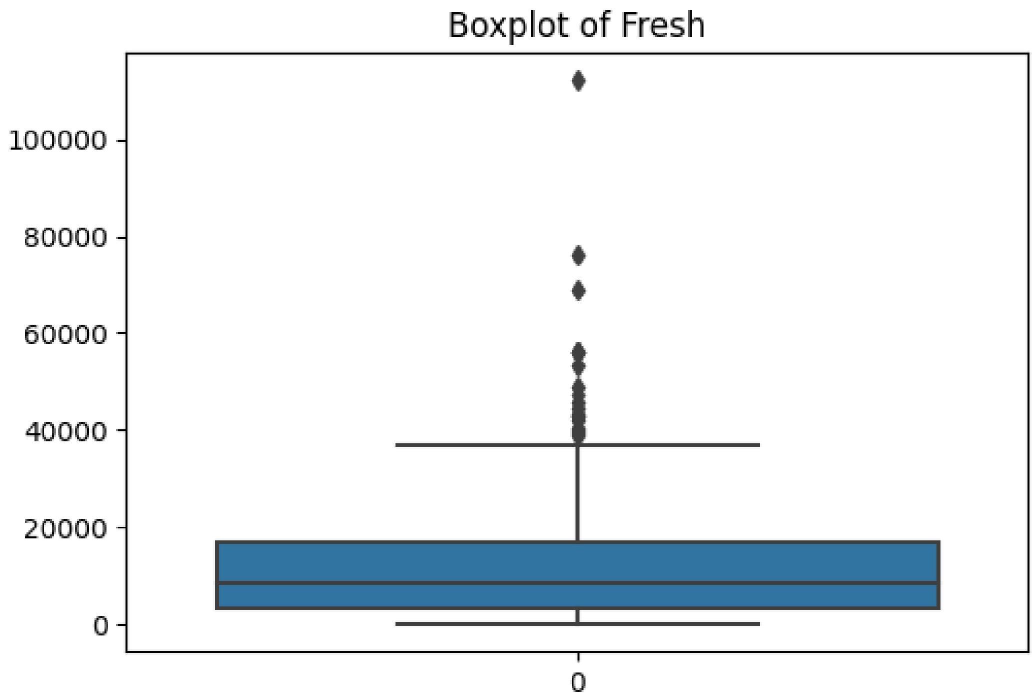
```
<Axes: >
```

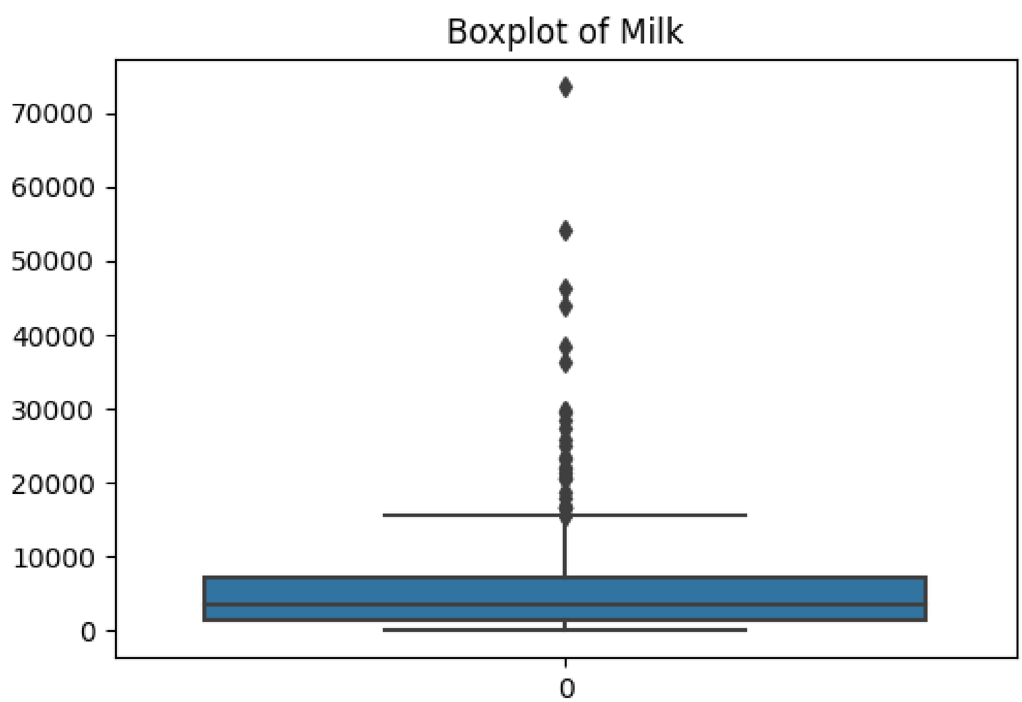


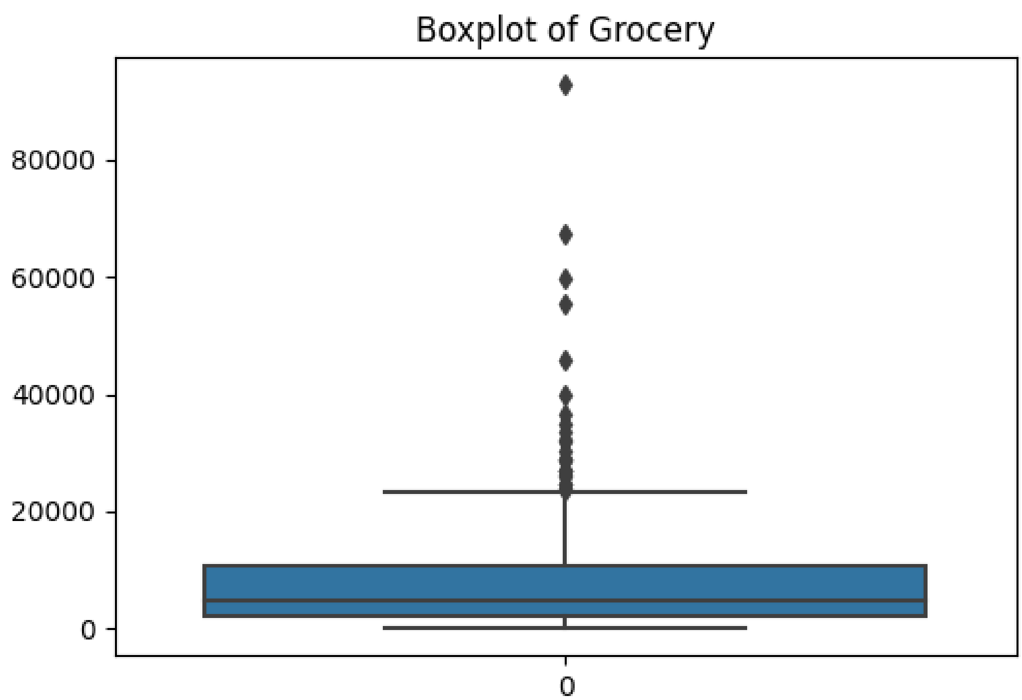

```
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.boxplot(df[column])
    plt.title(f'Boxplot of {column}')
    plt.show()
```

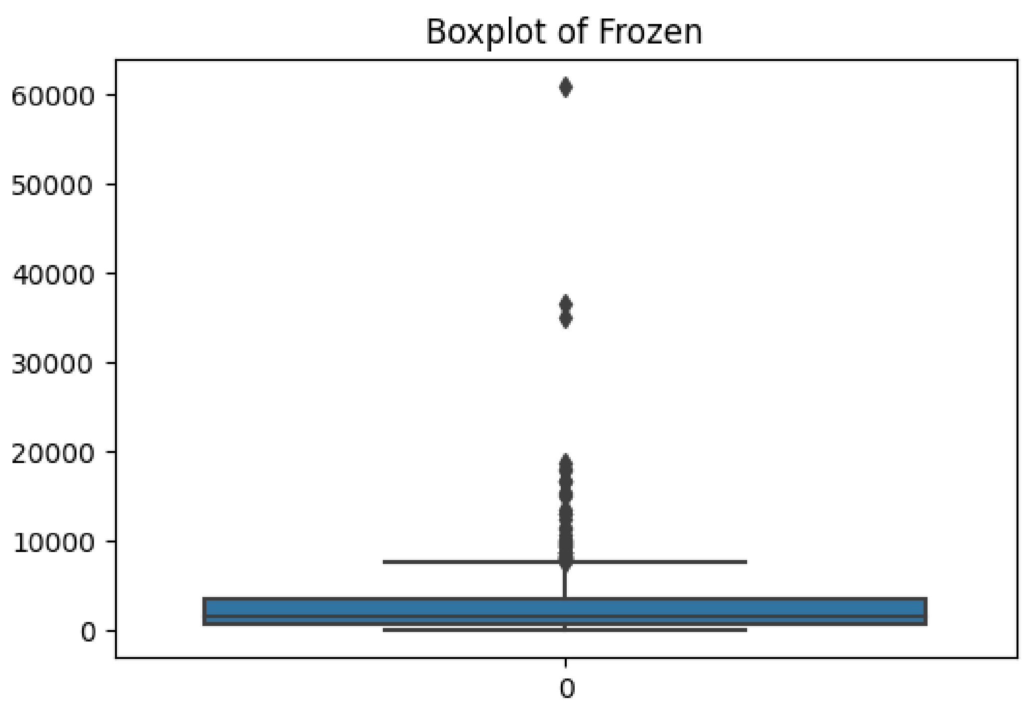

Boxplot of Channel
2.0
1.8
1.6
1.4
1.2
1.0
0

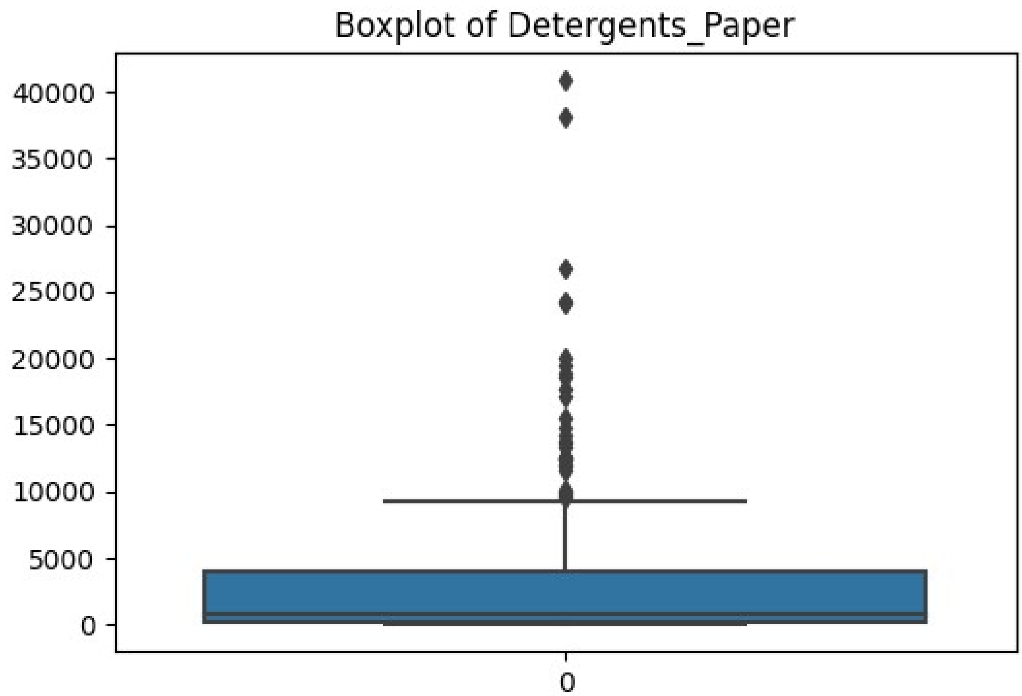

Boxplot of Region
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0


Boxplot of Fresh
100000
80000
60000
40000
20000
0
0


Boxplot of Milk
70000
60000
50000
40000
30000
20000
10000
0
0


Boxplot of Grocery
80000
60000
40000
20000
0
0


Boxplot of Frozen
60000
50000
40000
30000
20000
10000
0
0




```
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)

for column in df.columns:
    handle_outliers(df, column)
```

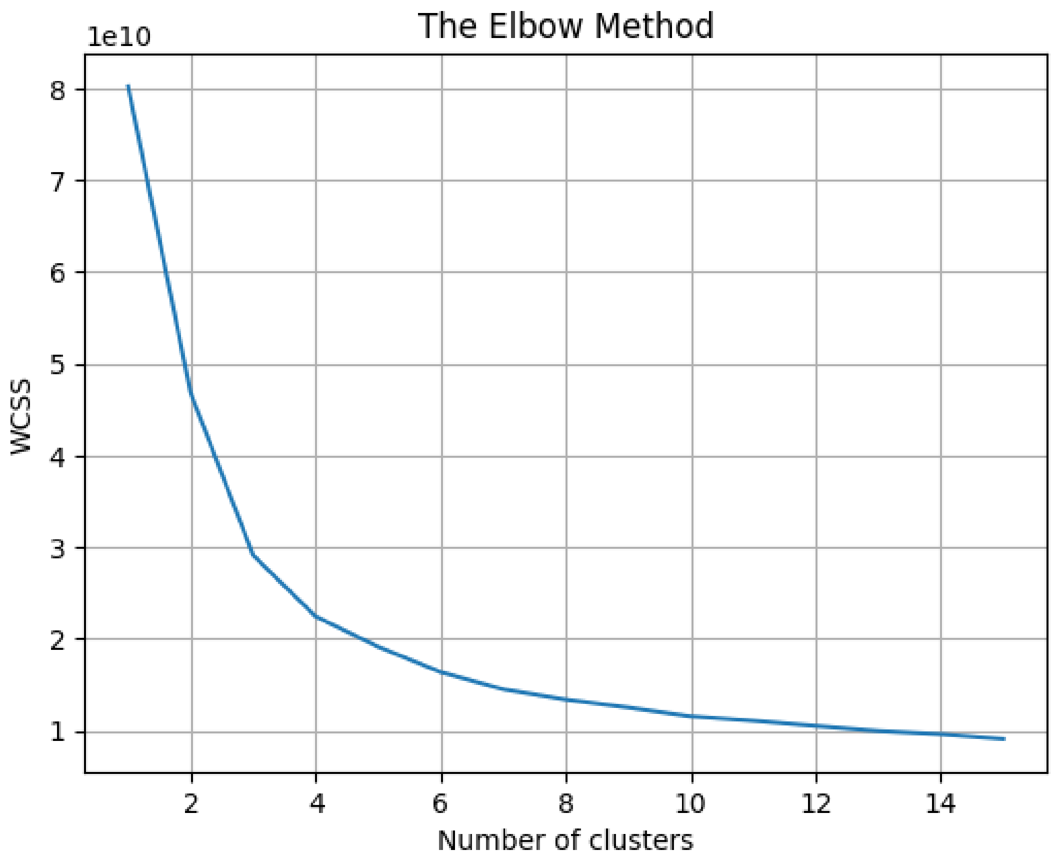
Machine Learning

```
from sklearn.preprocessing import StandardScaler

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

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

wcss = []
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_)

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(
```



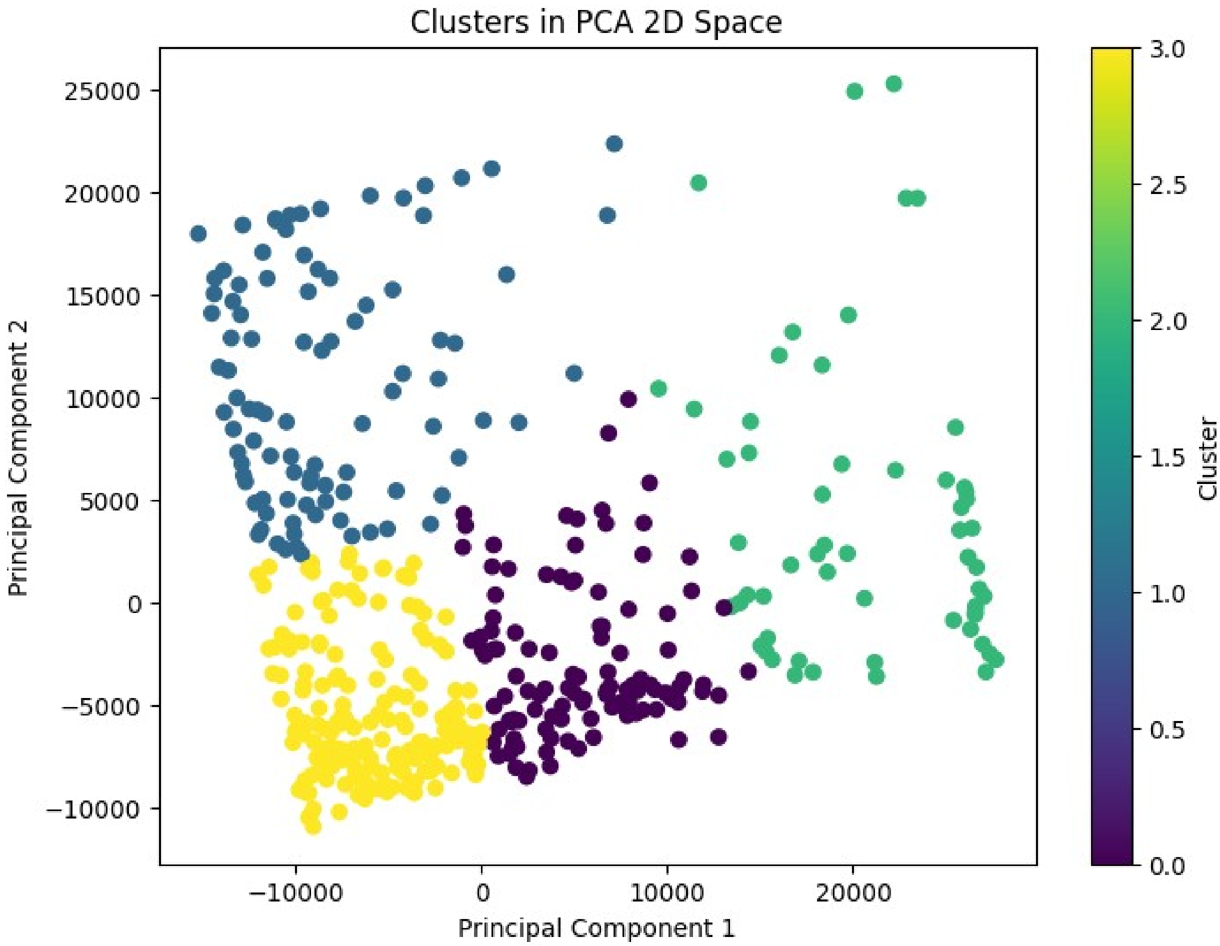

```
from sklearn.cluster import KMeans

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

cluster_labels = kmeans.labels_

df['Cluster'] = cluster_labels

print(df['Cluster'].unique())
```

```
/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
```

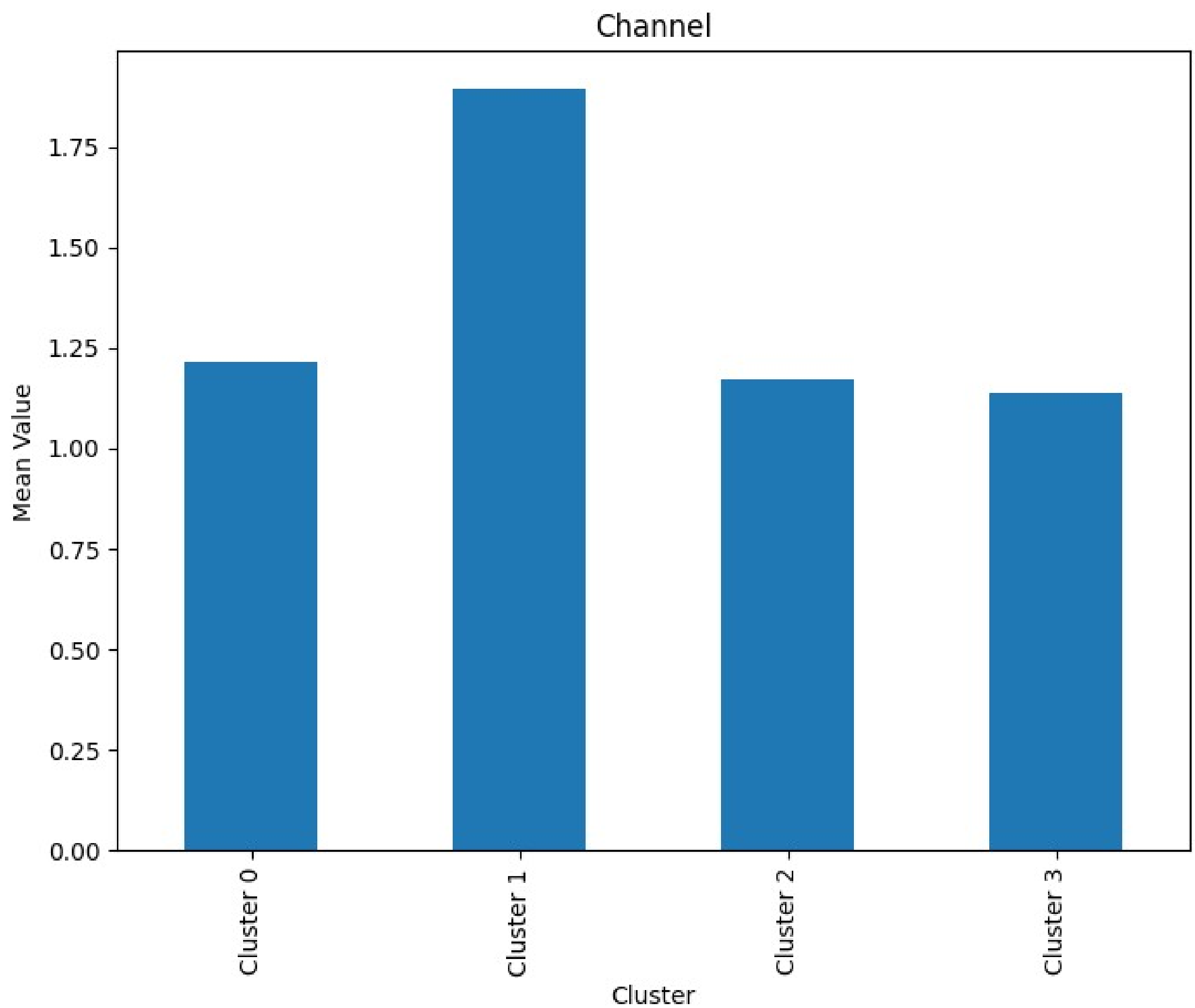
```
to suppress the warning
  warnings.warn(
```

```
[0 1 3 2]
```

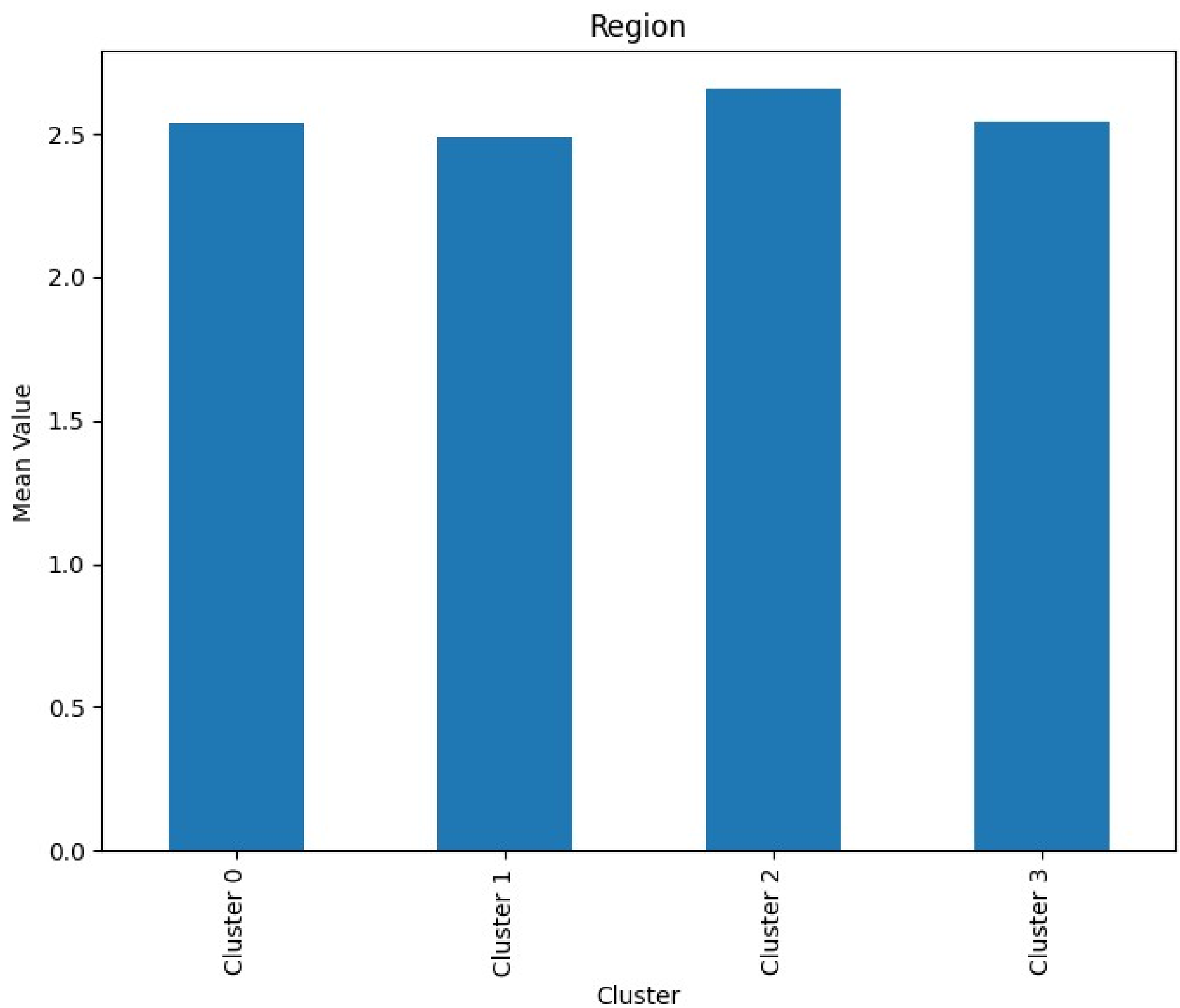
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df.drop('Cluster', axis=1))

PCA_components = pd.DataFrame(principalComponents, columns=['Principal
Component 1', 'Principal Component 2'])

PCA_components['Cluster'] = df['Cluster']

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()
```

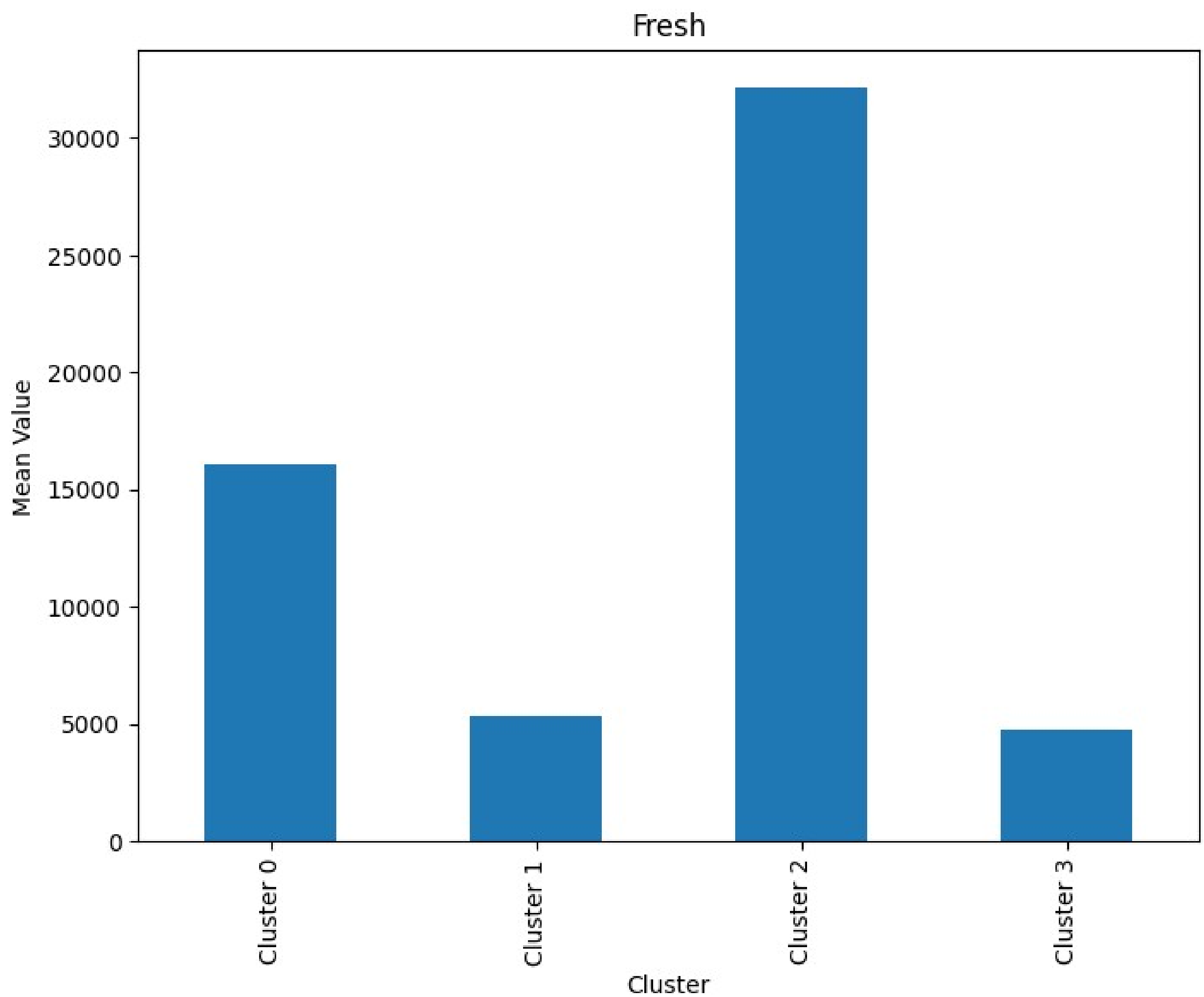


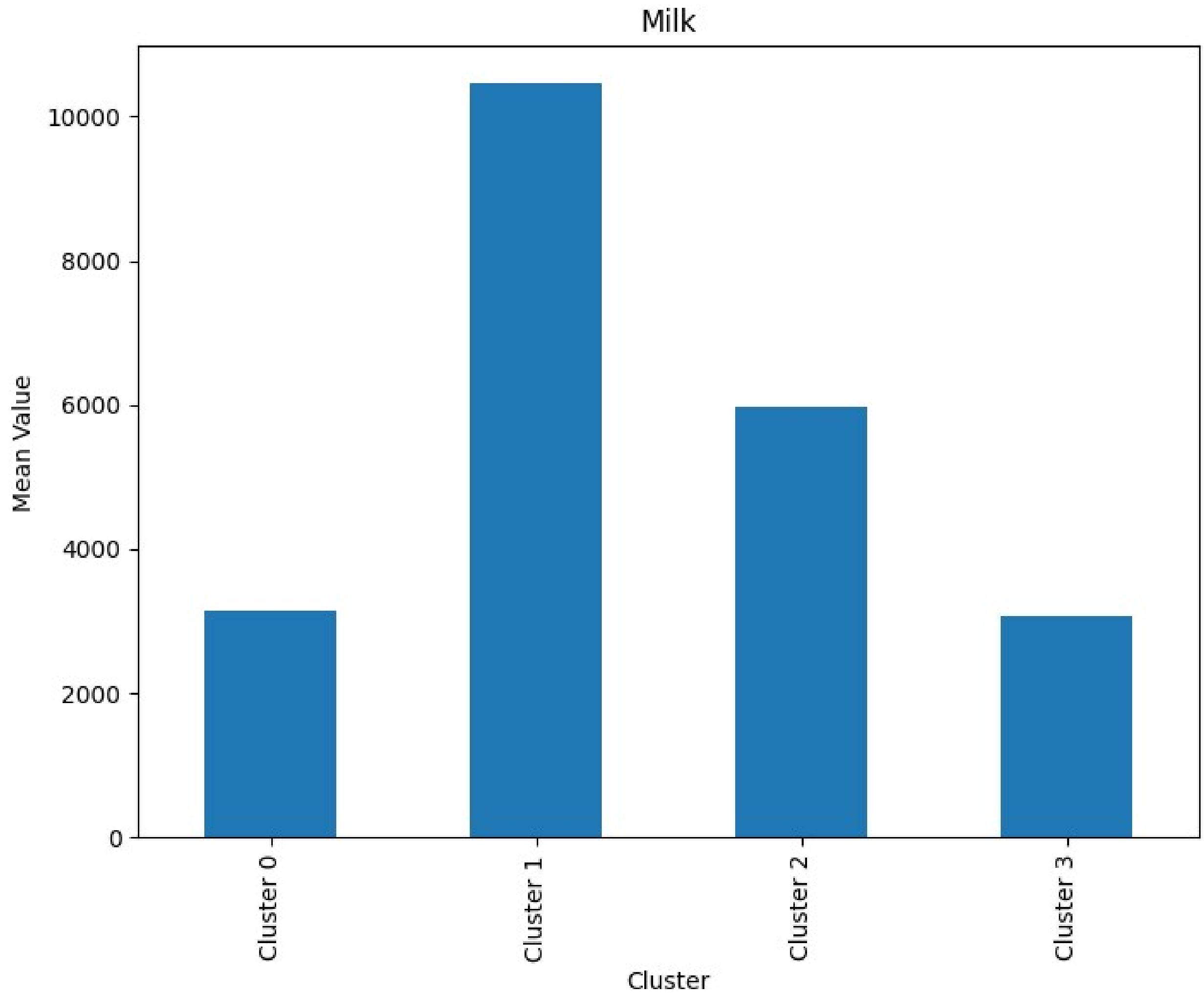

```python
cluster_means = df.groupby('Cluster').mean()

cluster_means = cluster_means.transpose()

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()
```

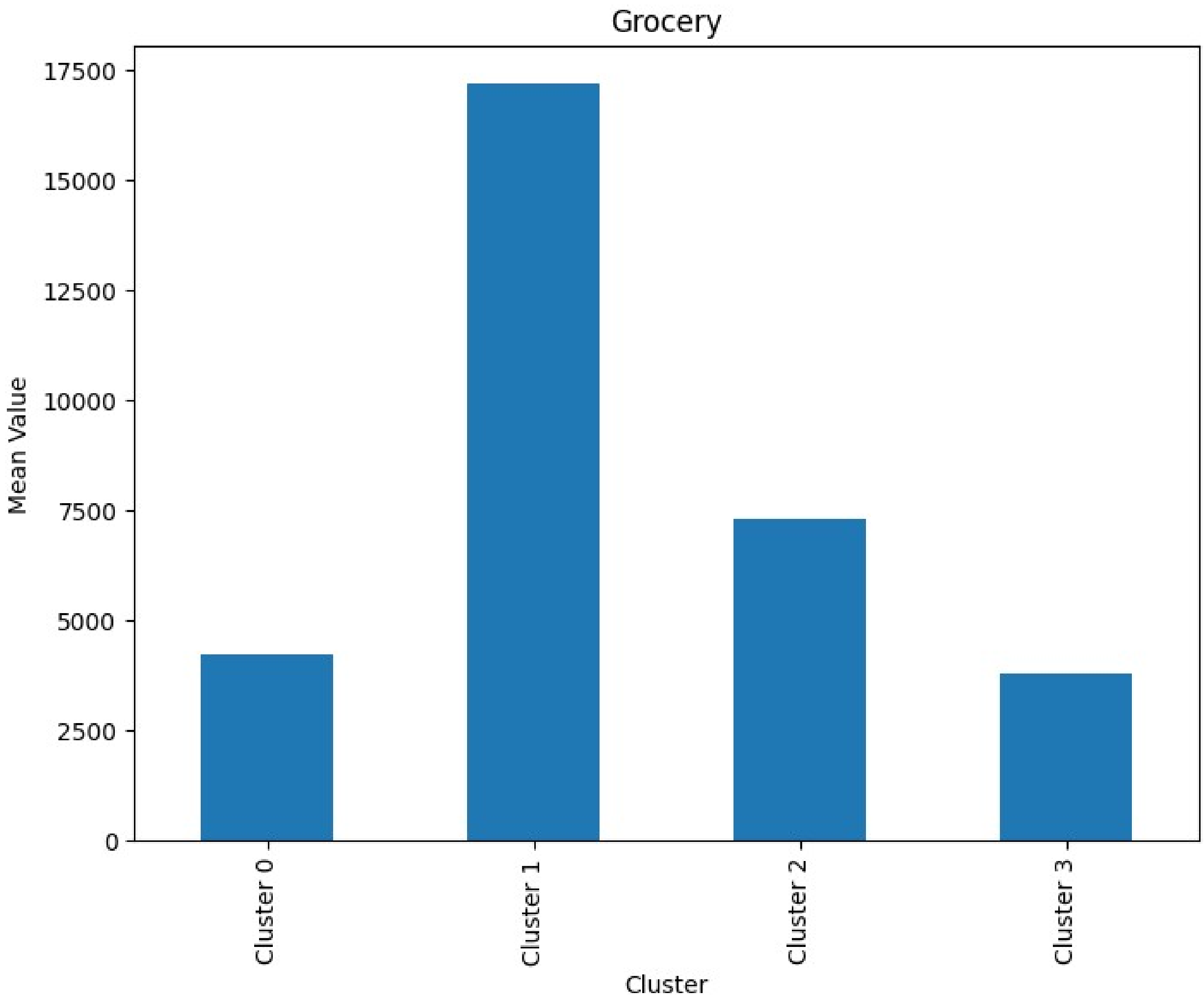


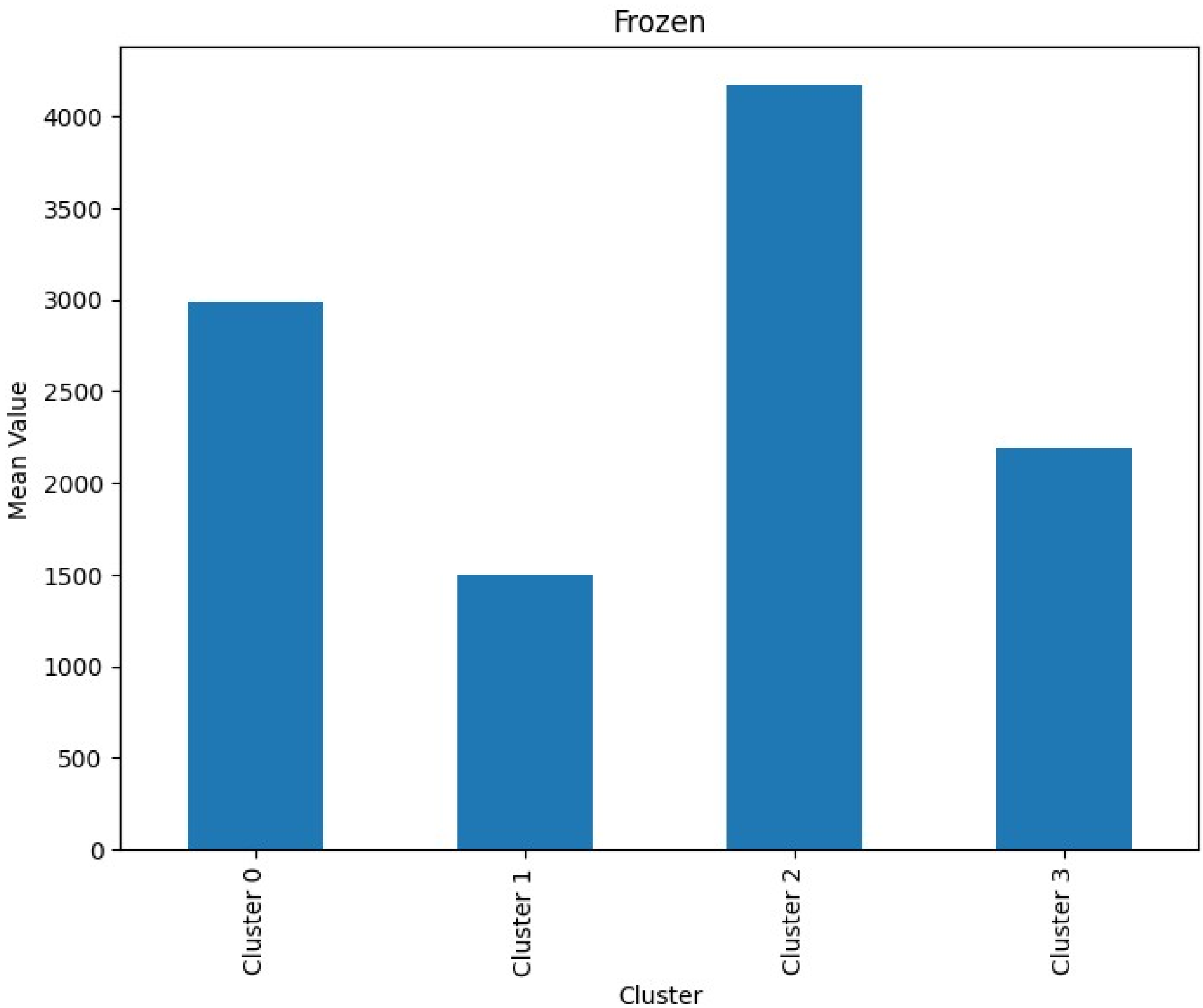
Channel
Mean Value
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster

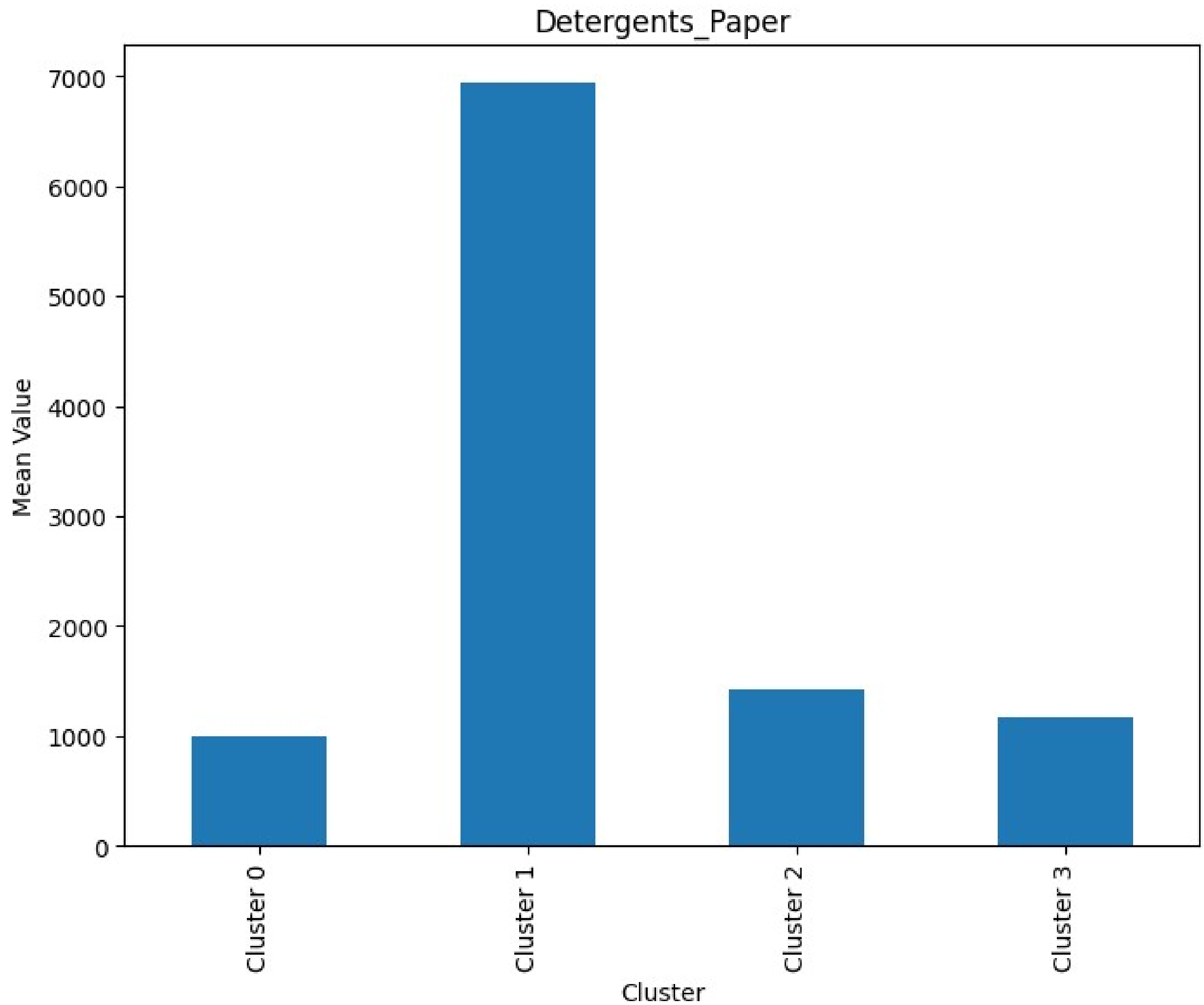

Region
Mean Value
0.0
0.5
1.0
1.5
2.0
2.5
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster

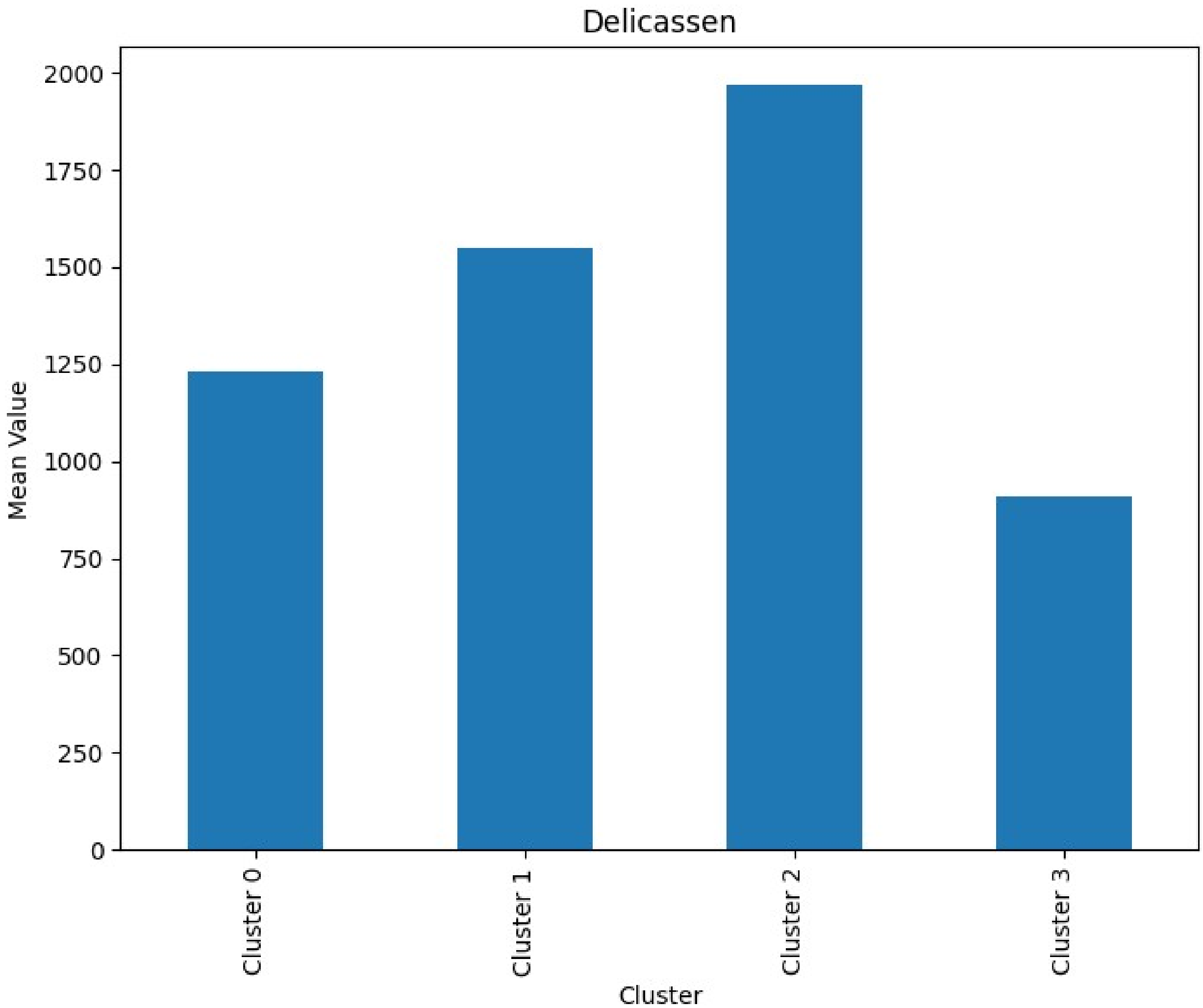

Fresh
Mean Value
30000
25000
20000
15000
10000
5000
0
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster


Milk
Mean Value
10000
8000
6000
4000
2000
0
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster


Grocery
17500
15000
12500
10000
7500
5000
2500
0
Mean Value
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster


Frozen
Mean Value
0
500
1000
1500
2000
2500
3000
3500
4000
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster


Detergents_Paper
Mean Value
7000
6000
5000
4000
3000
2000
1000
0
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster


Delicassen
Mean Value
2000
1750
1500
1250
1000
750
500
250
0
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster