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.

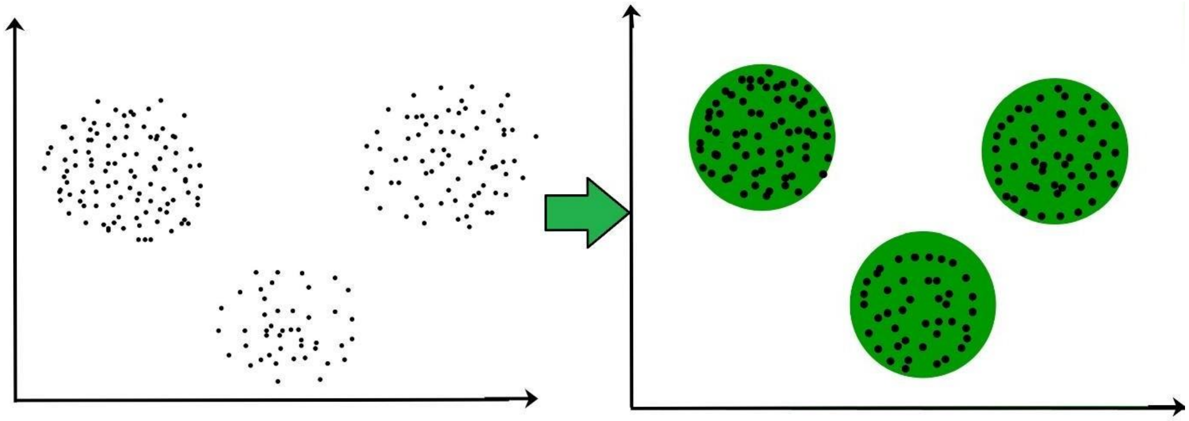
Objective: Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

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:

Conclusion :-

Using Clustered Data: Data clustering is vital for identifying distinct customer groups, enabling tailored strategies, optimizing operations, and boosting business success.

Cluster 0: "Diverse Shoppers" - Moderate purchases, balanced marketing.
Cluster 1: "Freshness Enthusiasts" - High demand for fresh, fast delivery.
Cluster 2: "Budget-Conscious Buyers" - Smaller purchases, cost-effective options.
Cluster 3: "High-Volume Demands" - Premium and efficient delivery for high-volume customers.

Adapting Delivery Schemes: Aligning delivery with customer preferences enhances satisfaction and business growth.

Cluster 0: Cost-effective, reliable options.
Cluster 1: Rapid, temperature-controlled delivery.
Cluster 2: Consolidated or slower delivery for cost reduction.
Cluster 3: Premium, bulk delivery for high-volume needs.

Importing Libs

```
# https://www.kaggle.com/code/ahmedhisham73/kmeans

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

```
df = pd.read_csv('/content/Wholesale.csv')
df.head(10)
```

	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
5	2	3	9413	8259	5126	666	1795	1451
6	2	3	12126	3199	6975	480	3140	545
7	2	3	7579	4956	9426	1669	3321	2566
8	1	3	5963	3648	6192	425	1716	750
9	2	3	6006	11093	18881	1159	7425	2098

Data exploration:

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

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

```
print("Data types:")
print(df.dtypes)
```

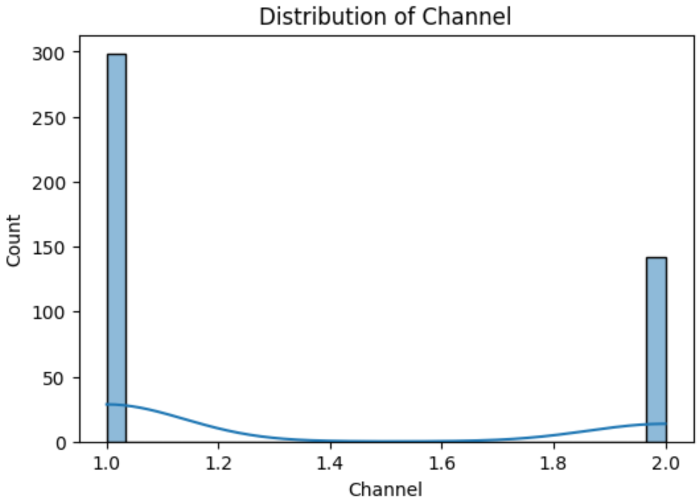
```
Data types:
Channel             int64
Region              int64
Fresh               int64
Milk                int64
Grocery             int64
Frozen              int64
Detergents_Paper    int64
Delicassen          int64
dtype: object
```

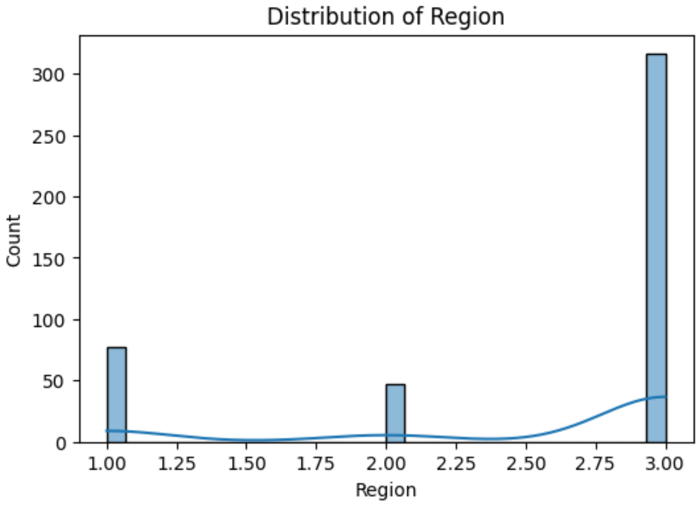
```
print("Missing values per column:")
print(df.isnull().sum())
```

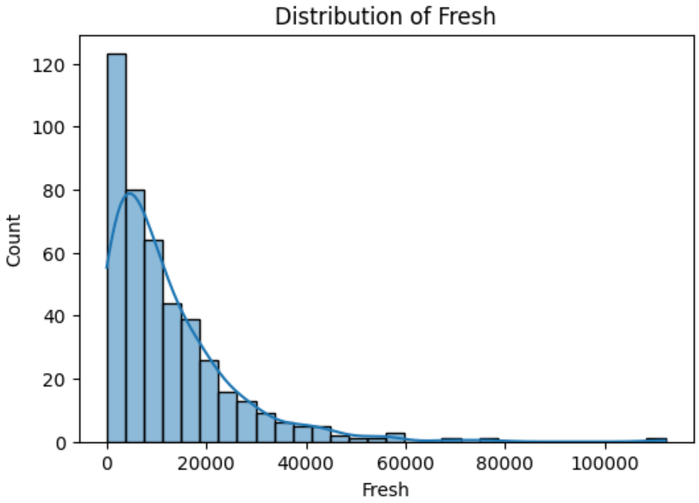
```
Missing values per column:
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
```

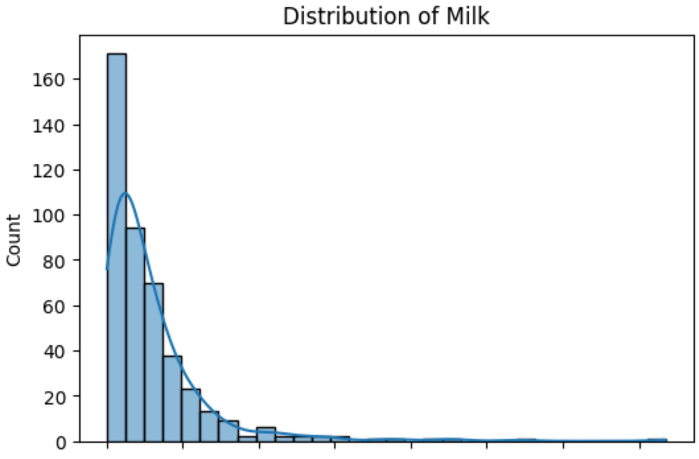
```
print("Descriptive Statistics:")
print(df.describe())

print("Number of duplicate rows: ", df.duplicated().sum())

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()
```

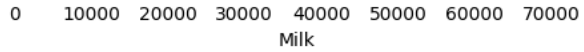
```
Descriptive Statistics:
          Channel      Region          Fresh          Milk       Grocery  \
count  440.000000  440.000000     440.000000    440.000000    440.000000
mean     1.322727    2.543182   12000.297727   5796.265909   7951.277273
std      0.468052    0.774272   12647.328865   7380.377175   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
```

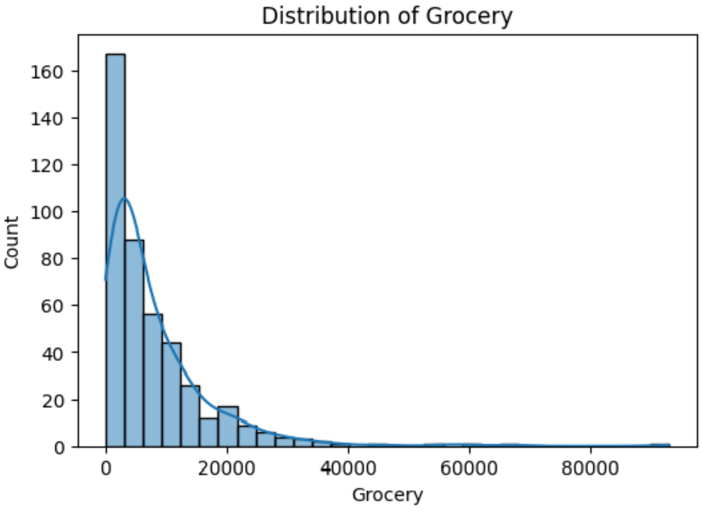

Distribution of Channel

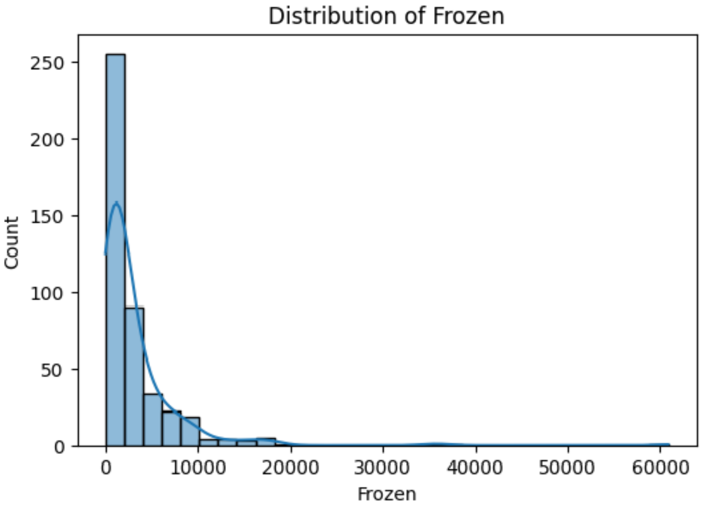

Distribution of Region

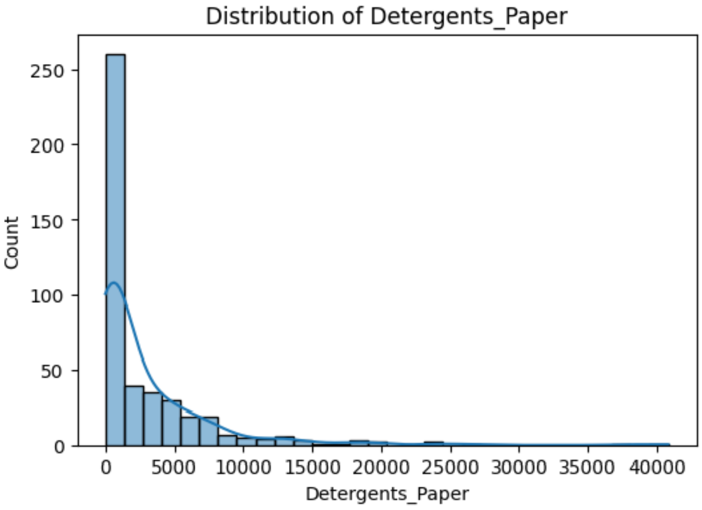

Distribution of Fresh

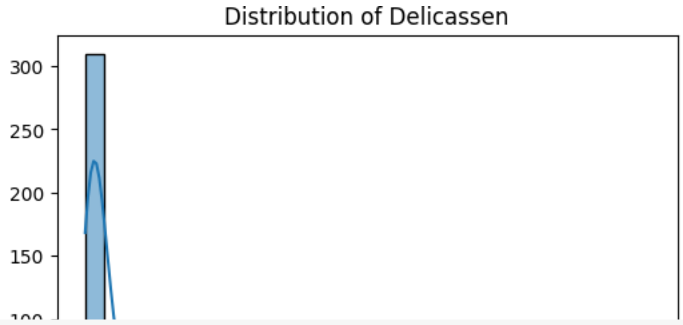

Distribution of Milk





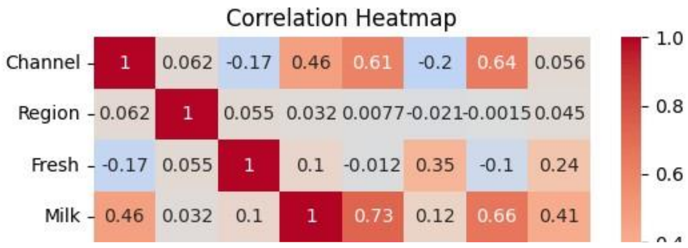




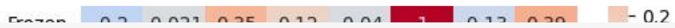


Heatmap

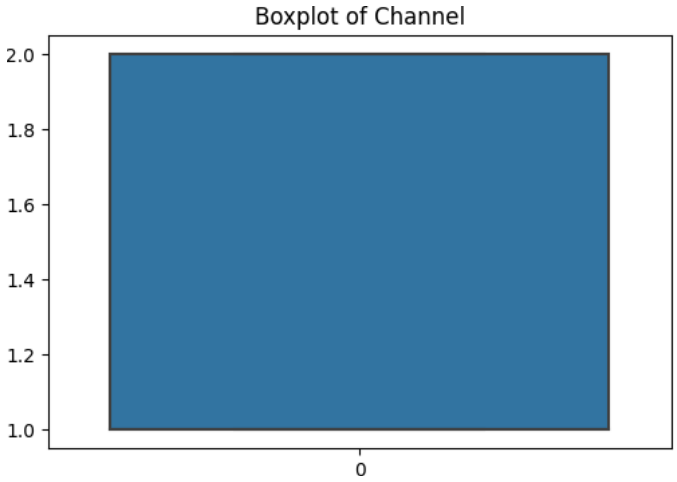
```
# Heatmap for correlation between variables
plt.figure(figsize=(6, 4))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap')
plt.show()
```

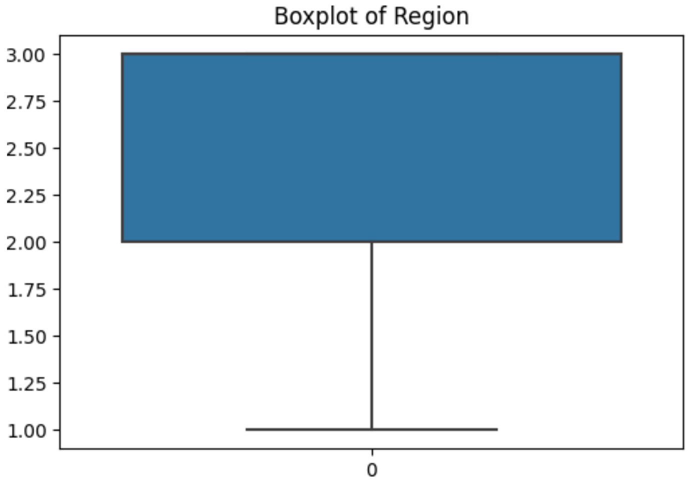


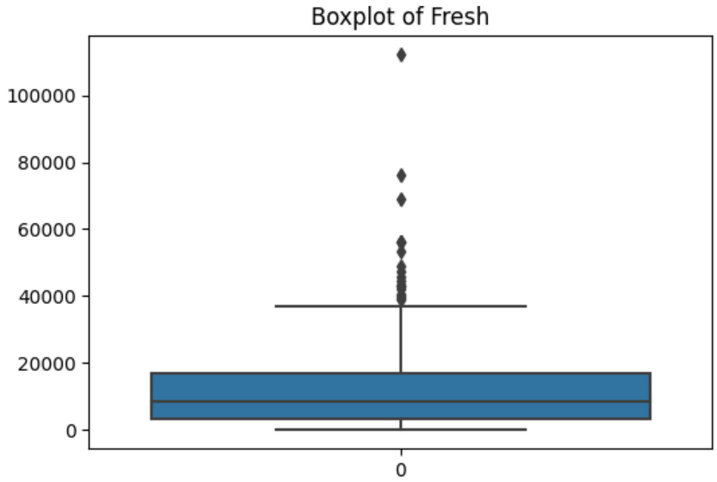
Checking for Outliers

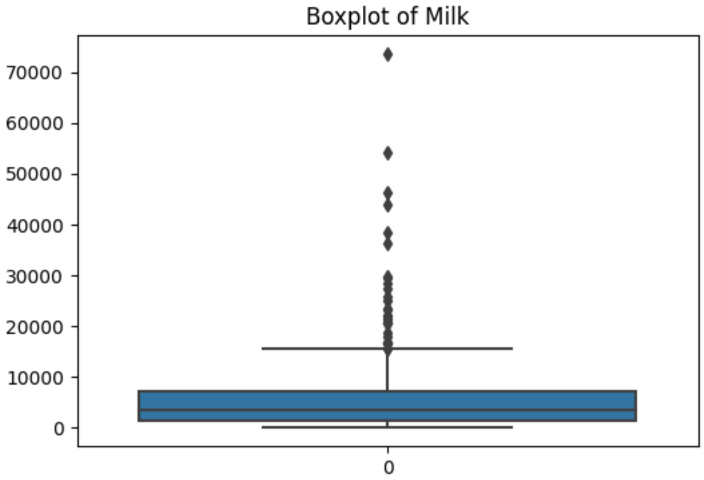


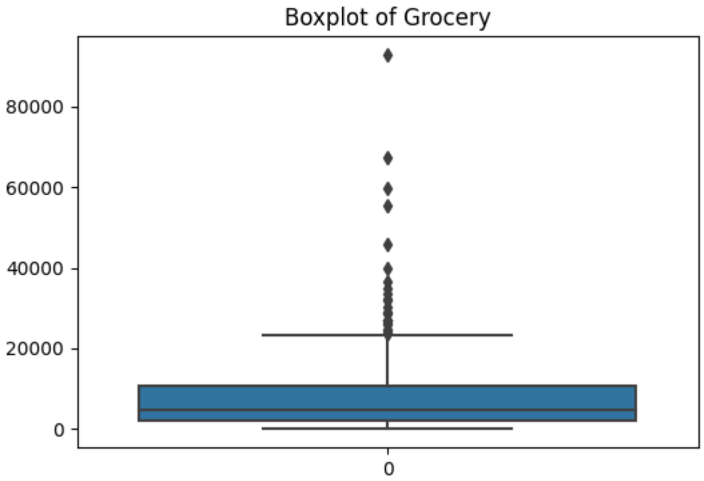
```
# Draw boxplots for all features
for column in df.columns:
    plt.figure(figsize=(6, 4))
    sns.boxplot(df[column])
    plt.title(f'Boxplot of {column}')
    plt.show()

# 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)}')
```









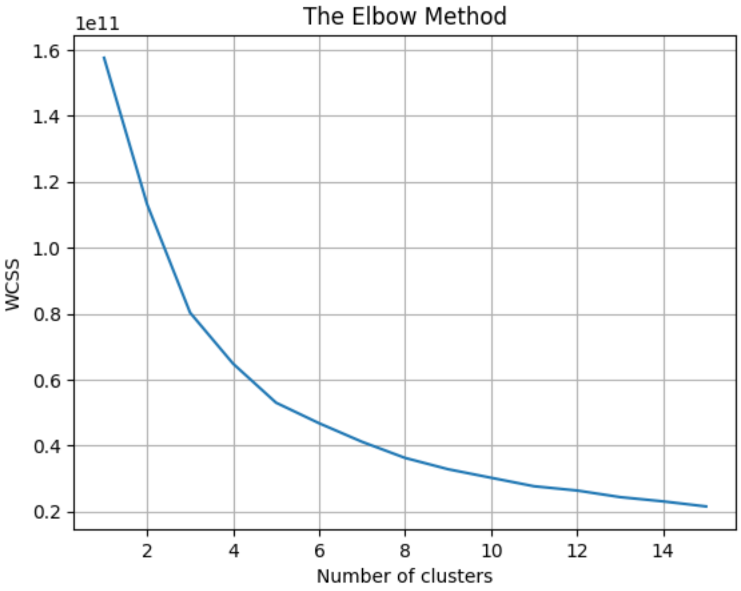


Boxplot of Frozen

Elbow Method to identify number of cluster Required

```
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

# Calculate WCSS for different number of clusters
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_)

# 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 d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The d
  warnings.warn(
```



KMeans

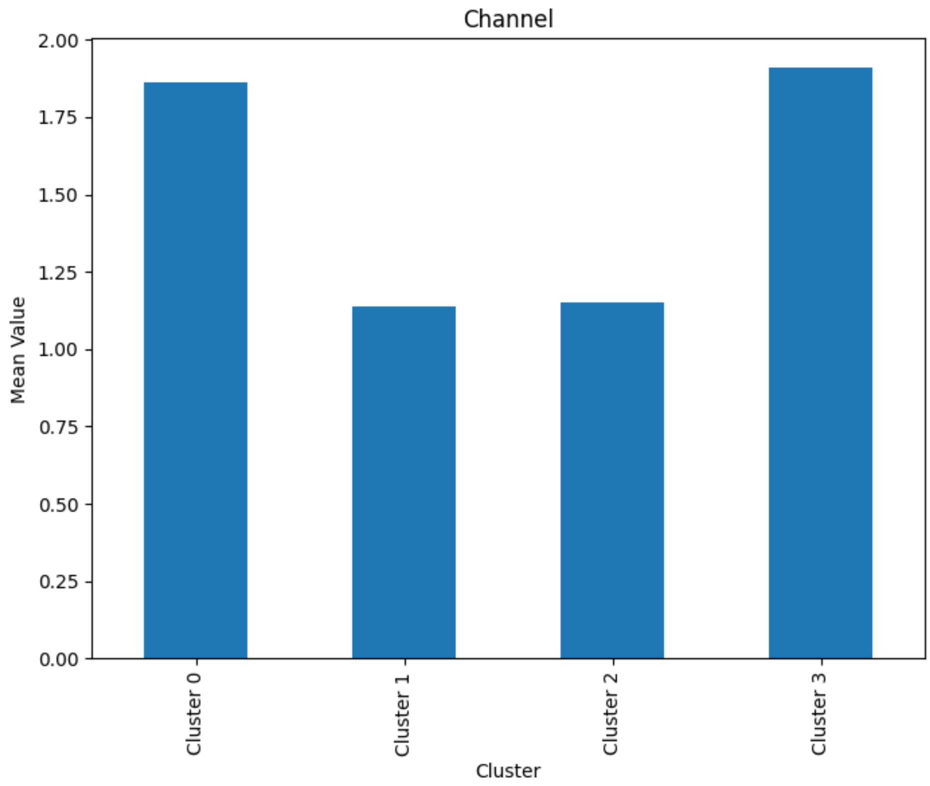
```
from sklearn.cluster import KMeans

# 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())
```

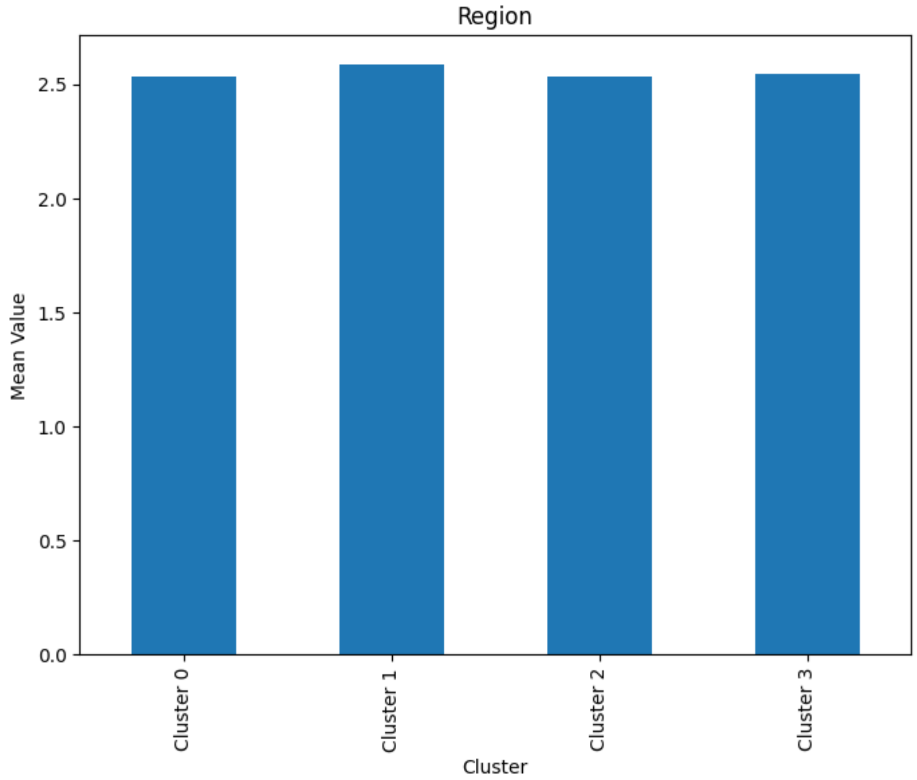
```
   Channel  Region  Fresh  Milk  Grocery  Frozen  Detergents_Paper  \
0        2       3  12669  9656     7561     214              2674
1        2       3   7057  9810     9568    1762              3293
2        2       3   6353  8808     7684    2405              3516
3        1       3  13265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

   Delicassen  Cluster
0        1338        2
1        1776        0
2        7844        2
3        1788        2
4        5185        1
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
```

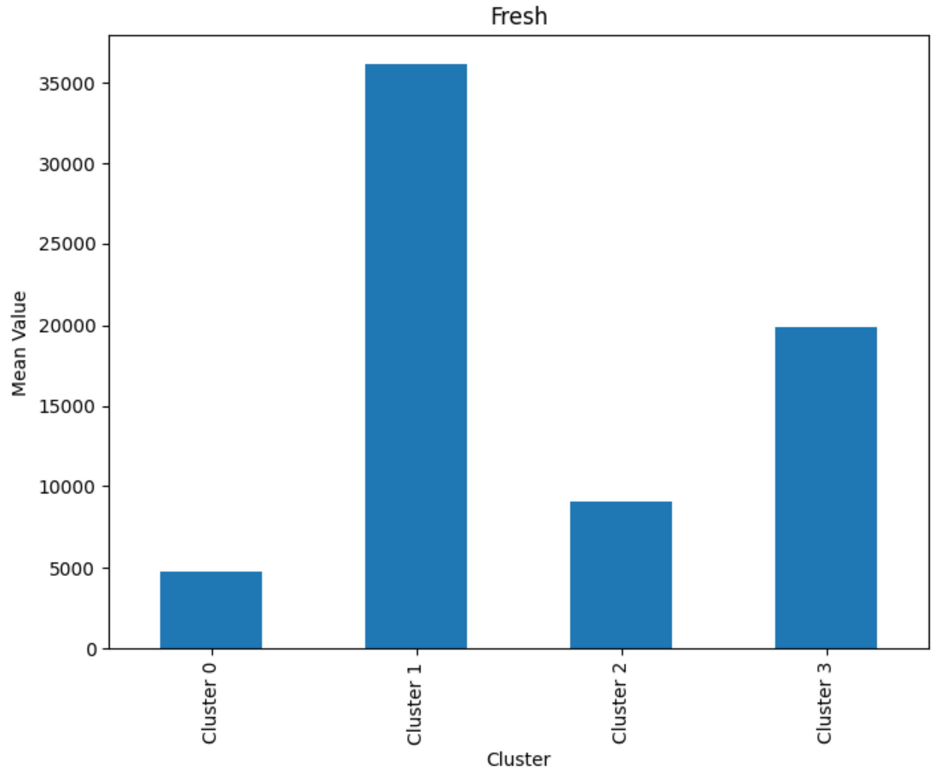
```
# Add cluster labels to the DataFrame
df['Cluster'] = kmeans.labels_

# 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())
```

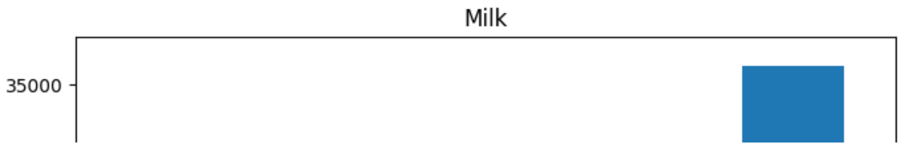
```
std     0.347839   0.773111   15406.720722   5059.930789   4289.538677
min     1.000000   1.000000   22096.000000    286.000000    471.000000
25%     1.000000   3.000000   26434.750000   2158.750000   2618.750000
50%     1.000000   3.000000   30818.000000   3954.500000   5058.500000
75%     1.000000   3.000000   40604.250000   7103.500000   8219.000000
max     2.000000   3.000000  112151.000000  29627.000000  18148.000000

             Frozen  Detergents_Paper    Delicassen  Cluster
count     58.000000         58.000000     58.000000     58.0
mean    6298.655172       1064.000000   2316.724138      1.0
std     8840.373423       1317.904703   2409.193705      0.0
min      127.000000         10.000000      3.000000      1.0
25%     1370.750000        284.000000    975.500000      1.0
50%     3662.000000        561.500000   1535.500000      1.0
75%     8674.000000       1135.500000   2798.000000      1.0
max    60869.000000       5058.000000  14351.000000      1.0

Cluster  2
          Channel      Region         Fresh          Milk       Grocery  \
count  276.000000  276.000000    276.000000    276.000000    276.000000
mean     1.152174    2.536232   9087.463768   3027.427536   3753.514493
std      0.359842    0.778431   6218.787958   2599.933332   2716.555045
min      1.000000    1.000000      3.000000     55.000000      3.000000
25%      1.000000    2.000000   3454.250000   1133.750000   1755.500000
50%      1.000000    3.000000   8257.500000   2193.000000   2849.000000
75%      1.000000    3.000000  13582.750000   4192.500000   5231.000000
max      2.000000    3.000000  23257.000000  18664.000000  13462.000000

             Frozen  Detergents_Paper   Delicassen  Cluster
count    276.000000        276.000000   276.000000    276.0
mean    2817.985507       1003.003623  1040.525362      2.0
std     3614.905029       1233.205498  1013.744595      0.0
min       47.000000          3.000000     3.000000      2.0
25%      779.000000        199.750000   360.750000      2.0
50%     1571.000000        436.000000   713.500000      2.0
75%     3505.250000       1322.750000  1417.250000      2.0
max    35009.000000       5316.000000  7844.000000      2.0

Cluster  3
         Channel     Region         Fresh          Milk       Grocery  \
count  11.000000  11.000000     11.000000     11.000000     11.000000
mean    1.909091   2.545455  19888.272727  36142.363636  45517.454545
std     0.301511   0.820200  14488.239473  18433.364784  21771.475666
min     1.000000   1.000000     85.000000   4980.000000  20170.000000
25%     2.000000   2.500000   8881.500000  25302.500000  32074.000000
50%     2.000000   3.000000  16117.000000  36423.000000  39694.000000
75%     2.000000   3.000000  31157.500000  45073.500000  57584.500000
max     2.000000   3.000000  44466.000000  73498.000000  92780.000000

             Frozen  Detergents_Paper    Delicassen  Cluster
count     11.000000         11.000000     11.000000     11.0
mean    6328.909091      21417.090909   8414.000000      3.0
std    10355.038662      12078.543310  13829.020486      0.0
min       36.000000        239.000000    903.000000      3.0
25%     1006.500000      18750.000000   1720.000000      3.0
50%     3254.000000      20070.000000   2944.000000      3.0
75%     5950.000000      25466.000000   5797.500000      3.0
max    36534.000000      40827.000000  47943.000000      3.0
```

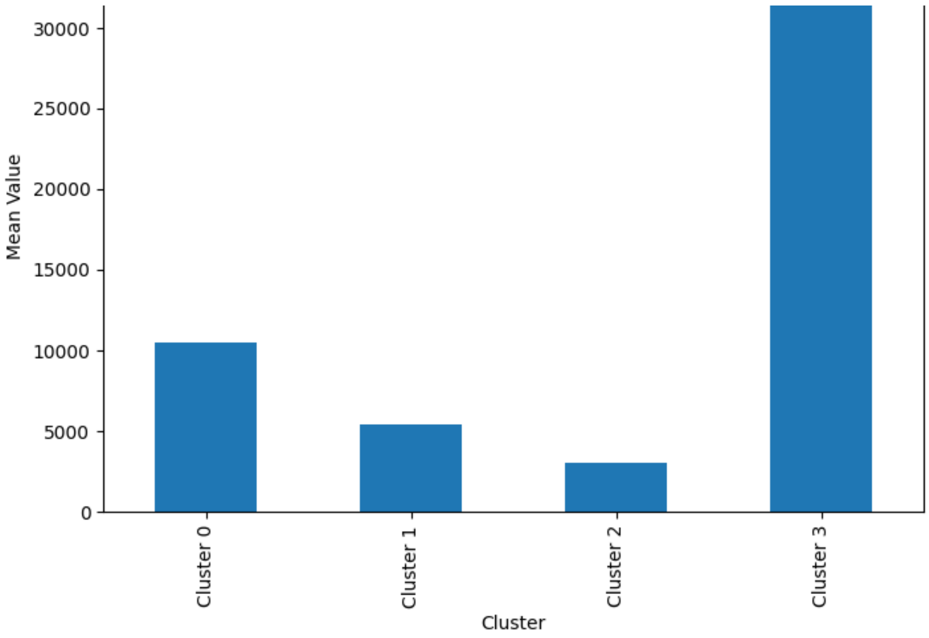
```
# Calculate the mean values for each feature per cluster
cluster_means = df.groupby('Cluster').mean()

# 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
Mean Value
2.00
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

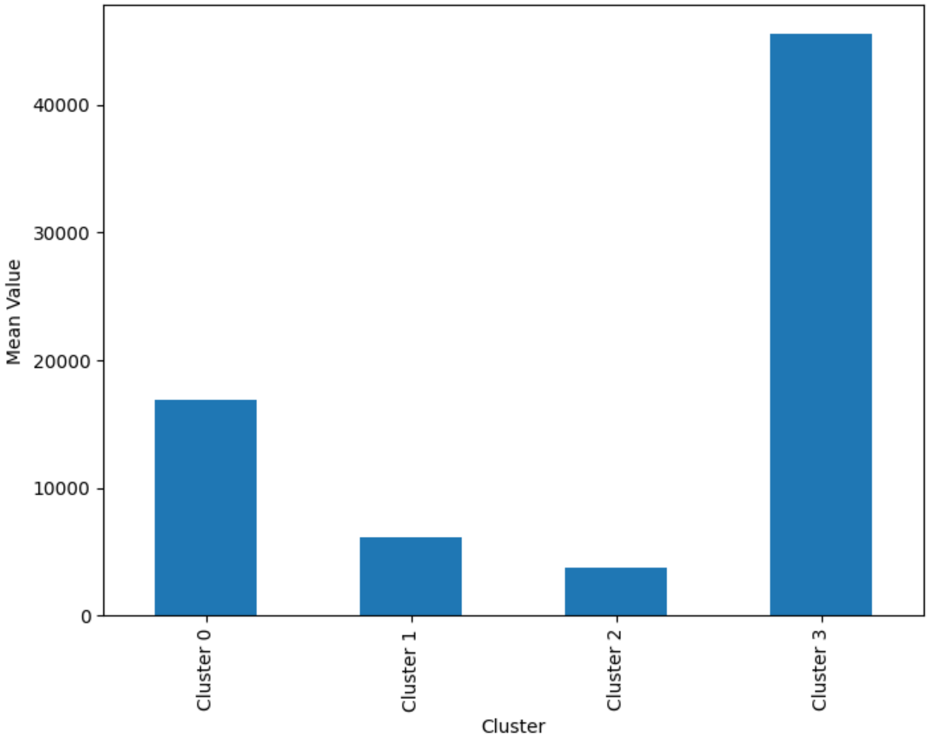

Region
Mean Value
2.5
2.0
1.5
1.0
0.5
0.0
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster


Fresh
Mean Value
35000
30000
25000
20000
15000
10000
5000
0
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster

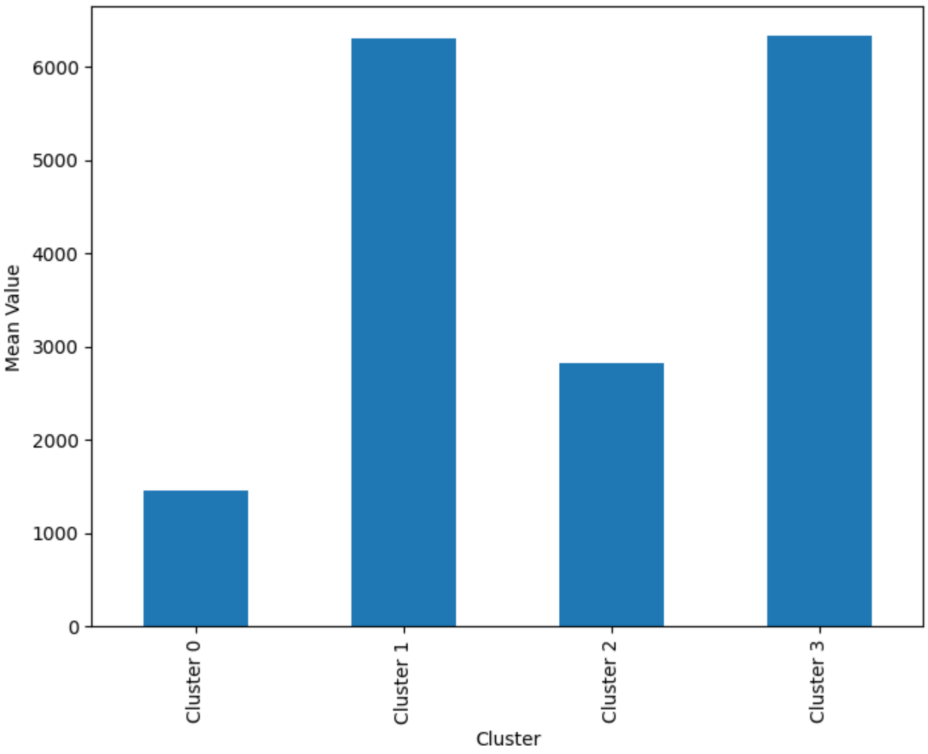

Milk
35000



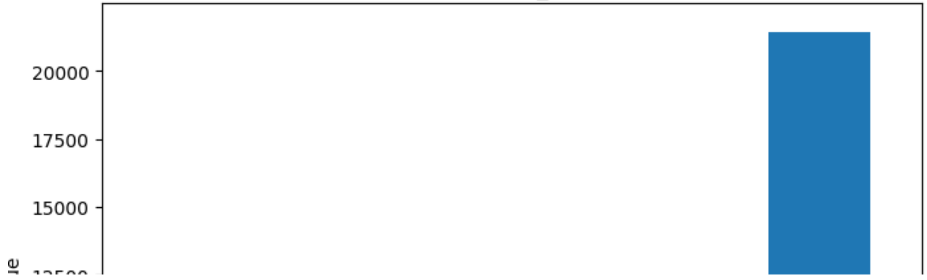
Grocery

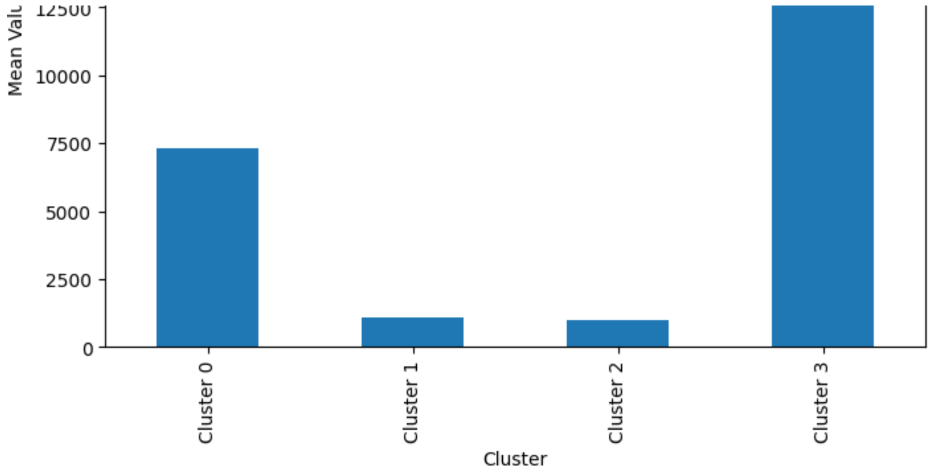


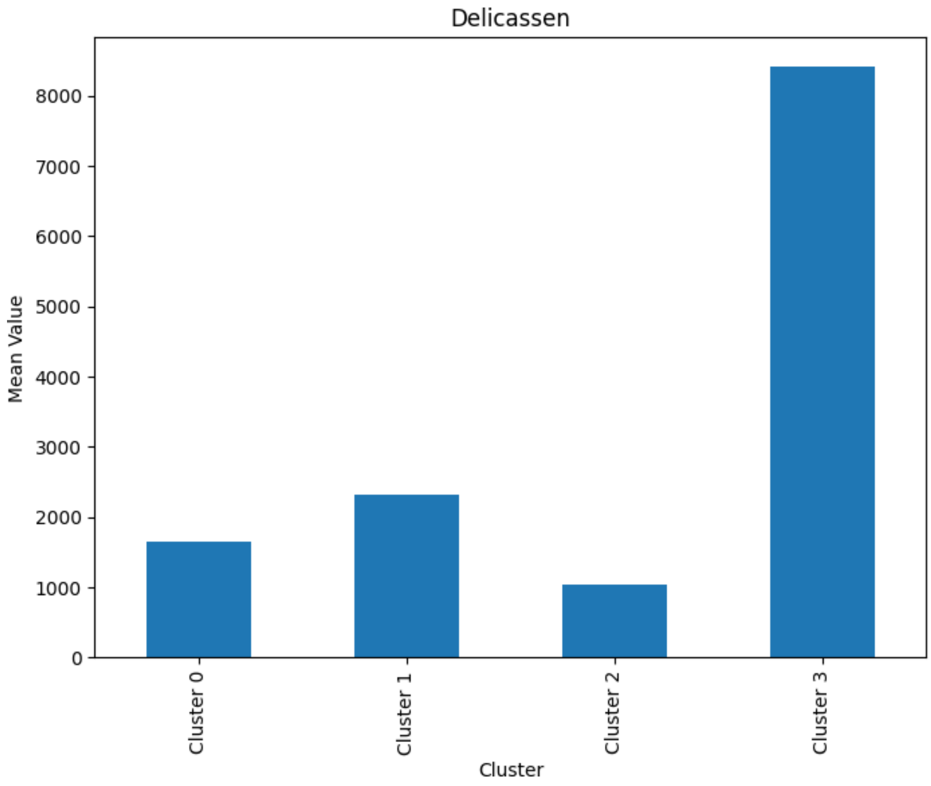
Frozen



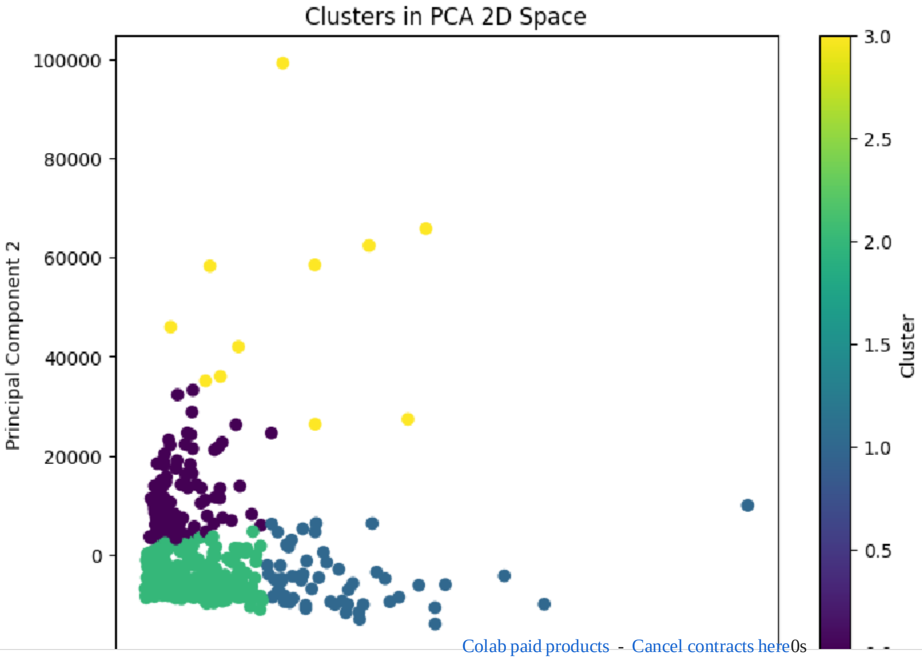
Detergents_Paper







```
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()
```



Colab paid products - Cancel contracts here0s

✓ completed at 12:37 AM