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.

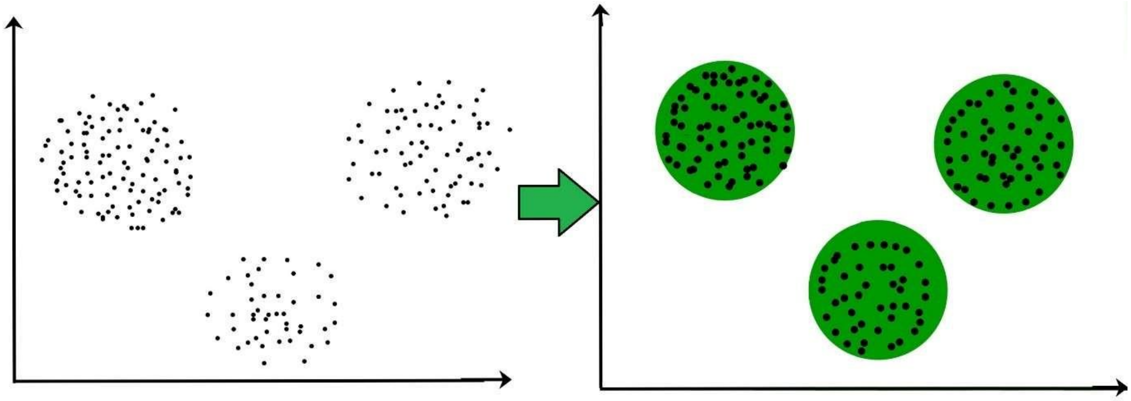
Objective: Able to perform various feature engineering tasks, apply Clustering Algorithm on the given dataset.

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)

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

```
# Load the dataset
data = pd.read_csv('Wholesale customers data.csv')
print(data)
```

```
     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
..       ...     ...    ...    ...      ...     ...               ...
435        1       3  29703  12051    16027   13135               182
436        1       3  39228   1431      764    4510                93
437        2       3  14531  15488    30243     437             14841
438        1       3  10290   1981     2232    1038               168
439        1       3   2787   1698     2510      65               477

     Delicassen
0          1338
1          1776
2          7844
3          1788
4          5185
..          ...
435        2204
436        2346
437        1867
438        2125
439          52

   [440 rows x 8 columns]
```

```
display(data.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

```
display(data.info())
```

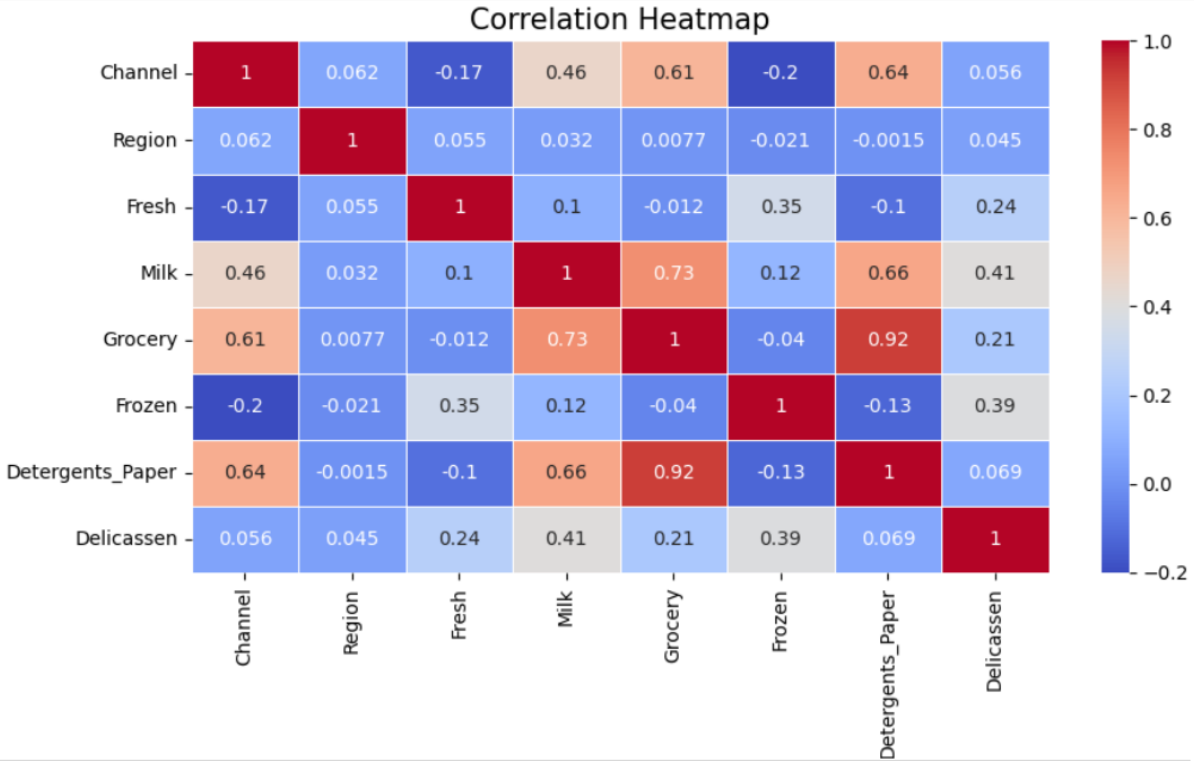
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Channel           440 non-null    int64
 1   Region            440 non-null    int64
 2   Fresh             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
None
```

```
display(data.describe())
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

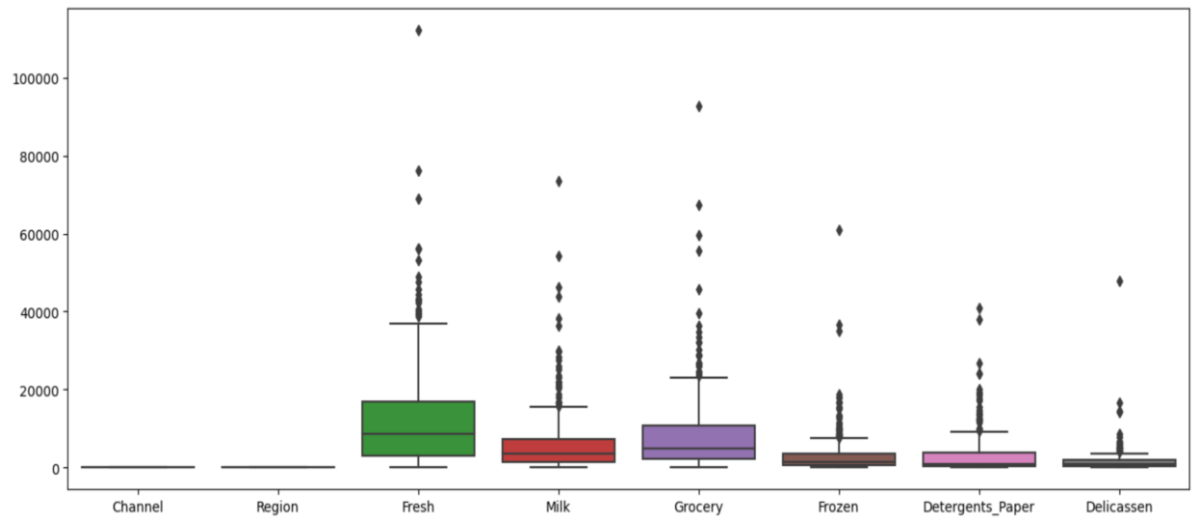
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix
corr = data.corr()

# Create a figure and axes for the heatmap
```

```
plt.figure(figsize=(10, 5))

# Create the heatmap
ax = sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)

# Set the title and adjust the fontsize
plt.title('Correlation Heatmap', fontsize=15)

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



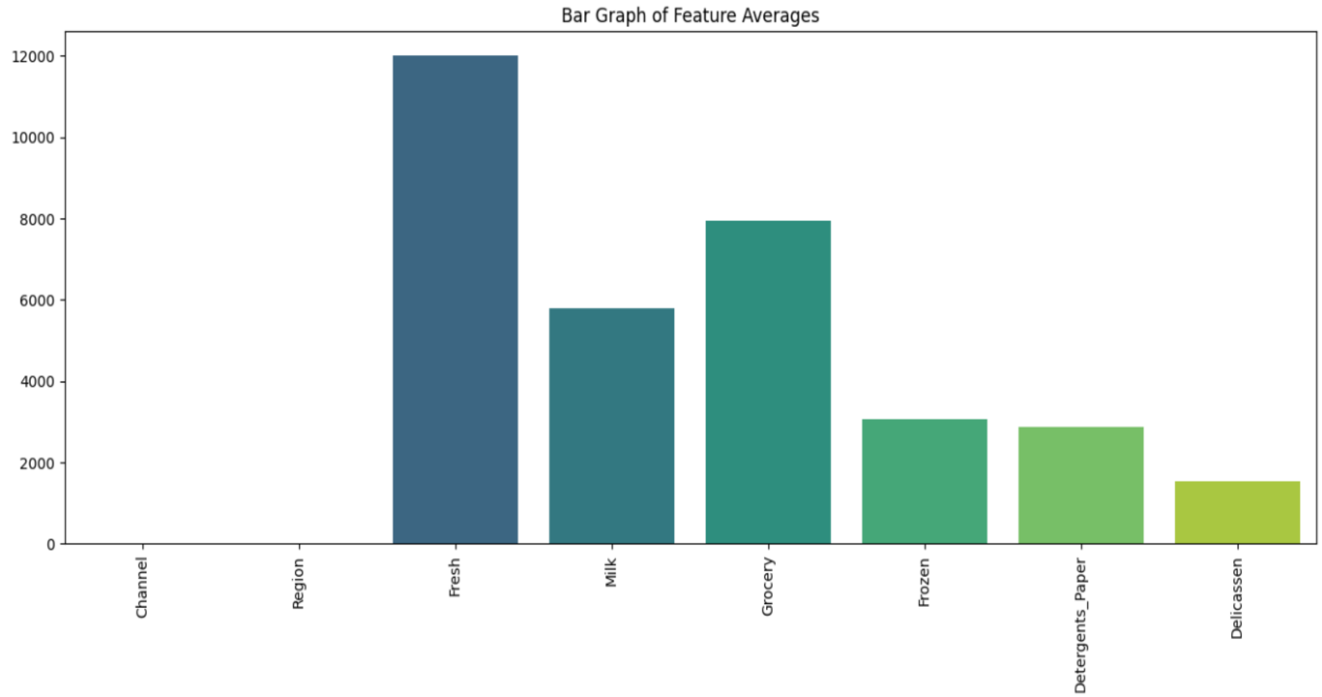
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler, RobustScaler,
StandardScaler, MinMaxScaler, PowerTransformer, MaxAbsScaler,
Normalizer, QuantileTransformer
import seaborn as sns
from sklearn import decomposition
import plotly.express as px

plt.figure(figsize=(16, 6))
sns.boxplot(data=data)
```

```
<Axes: >
```



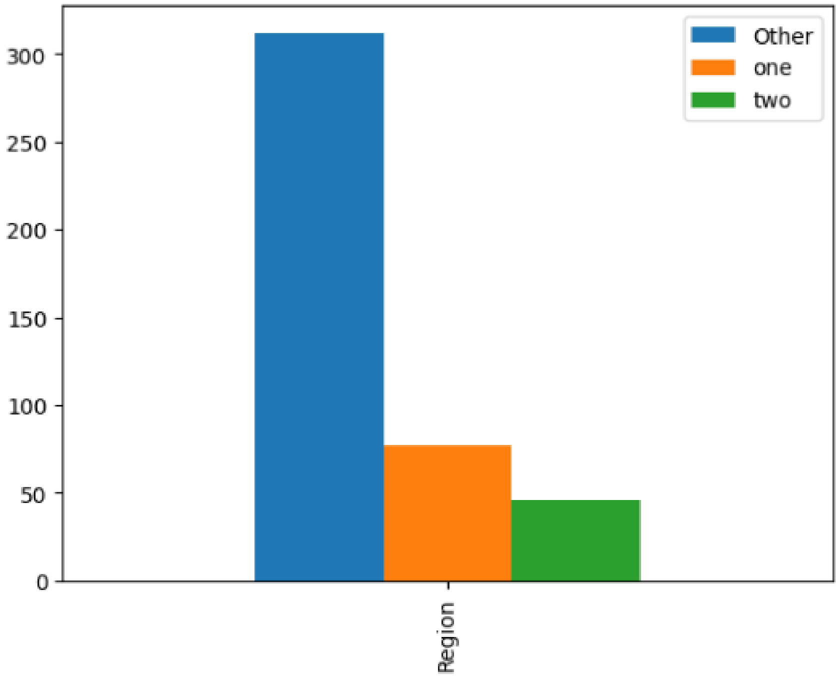
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming you have already loaded your data into 'data'

# Create a figure and axes
plt.figure(figsize=(16, 6))

# Plot a bar graph using the mean (average) values of each feature
sns.barplot(data=data, ci=None, palette='viridis')

# Set the title
plt.title('Bar Graph of Feature Averages')

# Rotate the x-axis labels for better visibility if needed
plt.xticks(rotation=90)

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

```
The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=data, ci=None, palette='viridis')
```

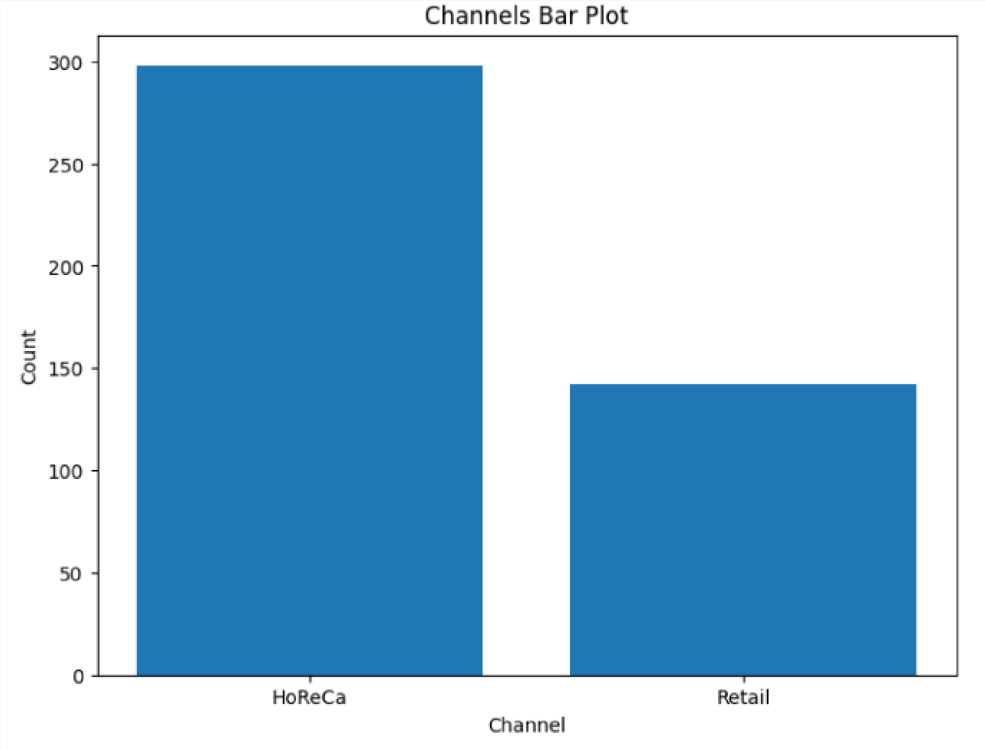


```
Region = pd.DataFrame(data['Region'].value_counts().T)
Region.rename(index={1:'one',2:'two',3:'Other'},inplace=True)
print('Region Bar Plot')
Region.T.plot.bar()
```

```
Region Bar Plot
<Axes: >
```

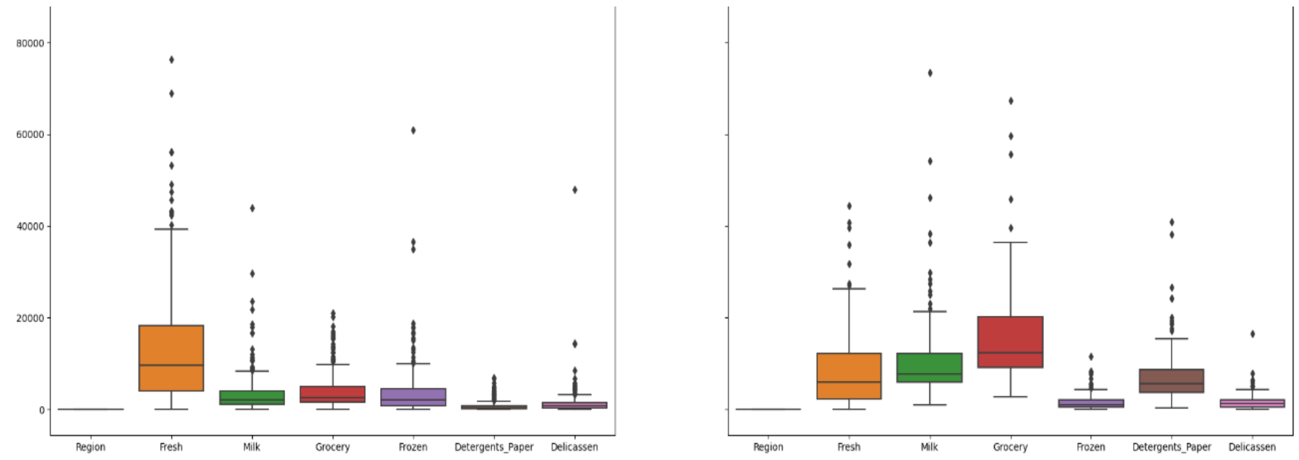


```
#drop region because it's very incomplete. It has a lot of "others"
df=data.drop(['Region'],axis=1)
import pandas as pd
import matplotlib.pyplot as plt

# Load the Wholesale Customers Dataset
data = pd.read_csv("Wholesale customers data.csv")

# Create a DataFrame from the value counts of the 'Channel' column
channel_counts = data['Channel'].value_counts().to_frame()

# Rename the index to 'HoReCa' and 'Retail'
channel_counts.rename(index={1: 'HoReCa', 2: 'Retail'}, inplace=True)

# Plot the bar chart
plt.figure(figsize=(8, 6))
plt.bar(channel_counts.index, channel_counts['Channel'])
plt.title('Channels Bar Plot')
plt.xlabel('Channel')
plt.ylabel('Count')
plt.show()
```

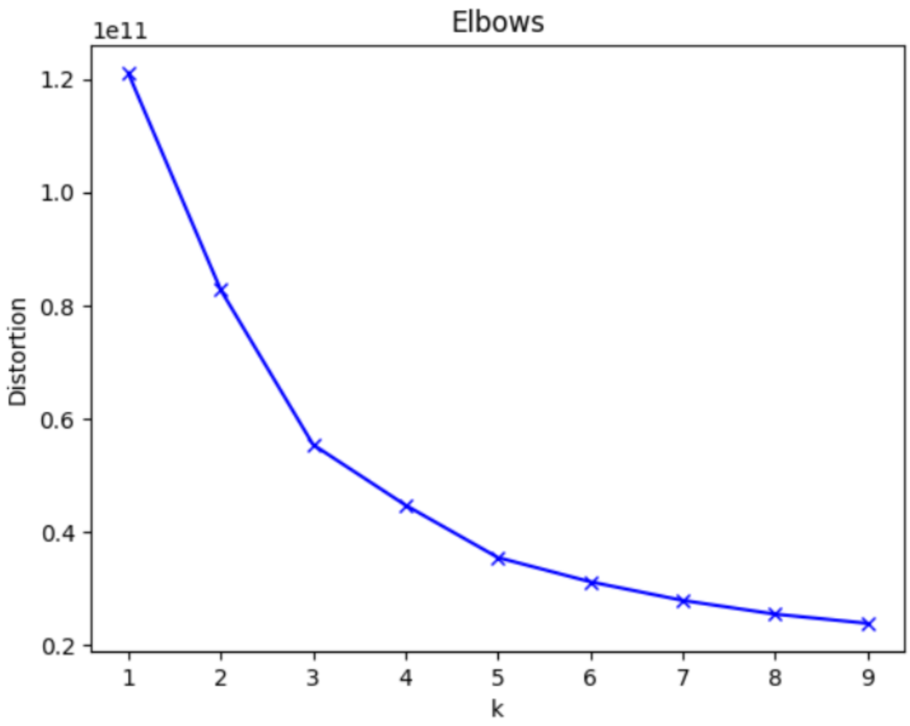


```
# Divide Retail from HoReCa and try to divide HoReCa in Hotel,
Restaurant and Café
dfHoReCa = data[data['Channel']==1].drop(['Channel'],axis=1)
```

```
dfRetail = data[data['Channel']==2].drop(['Channel'],axis=1)

# Plot both groups to visualize the difference
fig, (ax1, ax2) = plt.subplots(ncols=2, sharey=True,figsize=(25, 10))
ax1.set_title('Hotels / Restaurants / Cafés')
ax2.set_title('Retail')
sns.boxplot(data=dfHoReCa, ax=ax1)
sns.boxplot(data=dfRetail, ax=ax2)
```



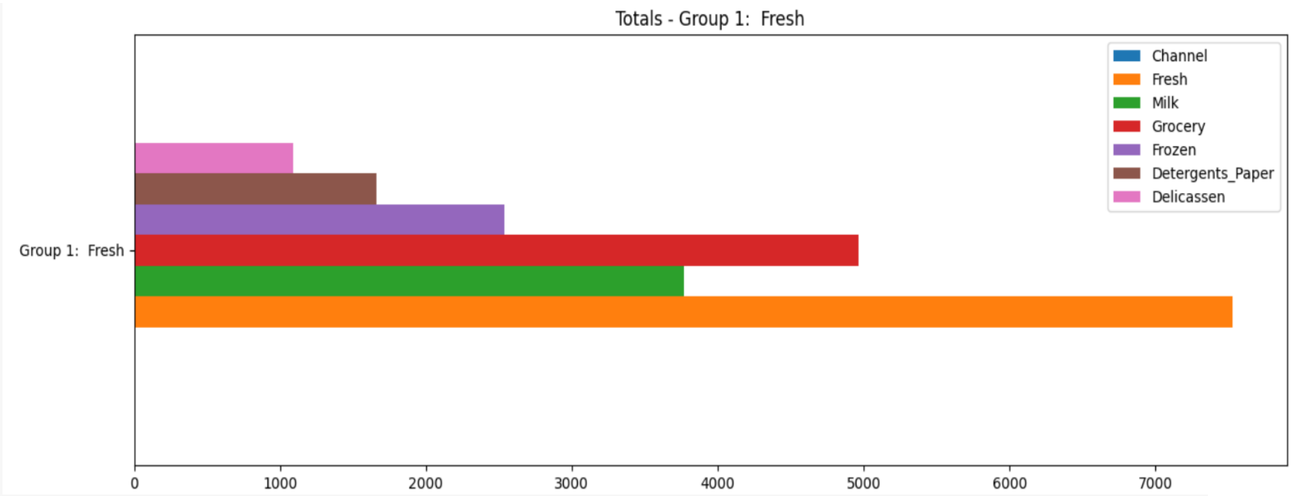
```
# Drop any unnecessary columns (if any)
data.drop(['Channel', 'Region'], axis=1, inplace=True)

# Standardize the data (important for some clustering algorithms)
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
#Full Dataset Analysis and Clustering

distortions = []
K = range(1,10)
for k in K:
    model = KMeans(n_clusters=k)
    model.fit(df)
    distortions.append(model.inertia_)
print(distortions)
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('Elbows')
plt.show()
```



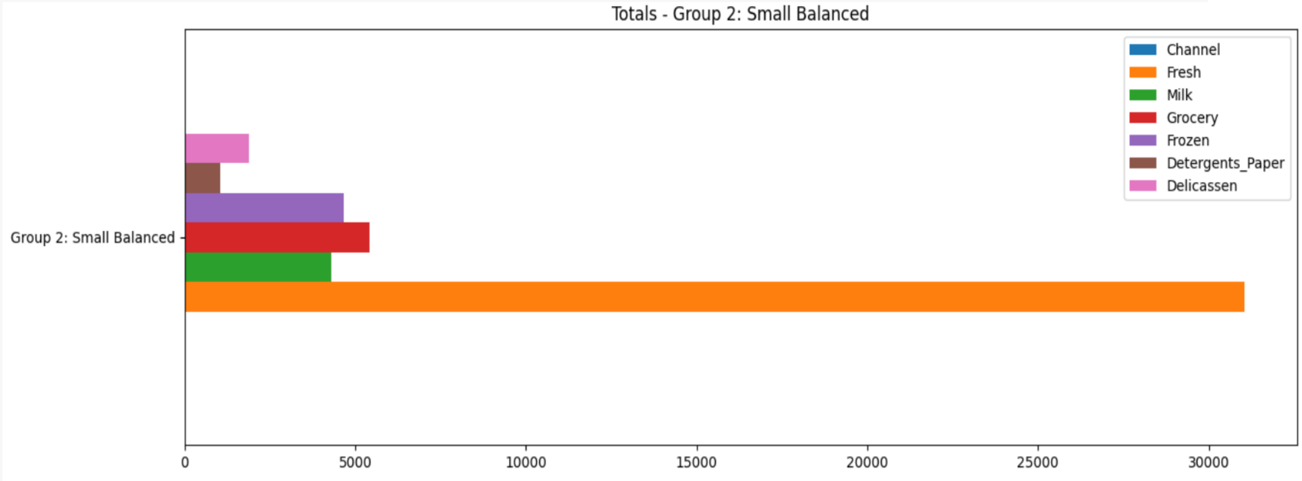
```
kmeans = KMeans(n_clusters=3,max_iter=1000,random_state=42)
kmeans.fit(df)
predict = kmeans.predict(df)
centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[1.24584718e+00 7.52670432e+03 3.77226578e+03 4.96659801e+03
  2.53832558e+03 1.66336877e+03 1.08719269e+03]
 [1.16883117e+00 3.10474935e+04 4.28449351e+03 5.41632468e+03
  4.65564935e+03 1.04909091e+03 1.88788312e+03]
 [1.92982456e+00 7.07331579e+03 1.53498772e+04 2.47319474e+04
  1.98189474e+03 1.08604737e+04 2.32249123e+03]]
```

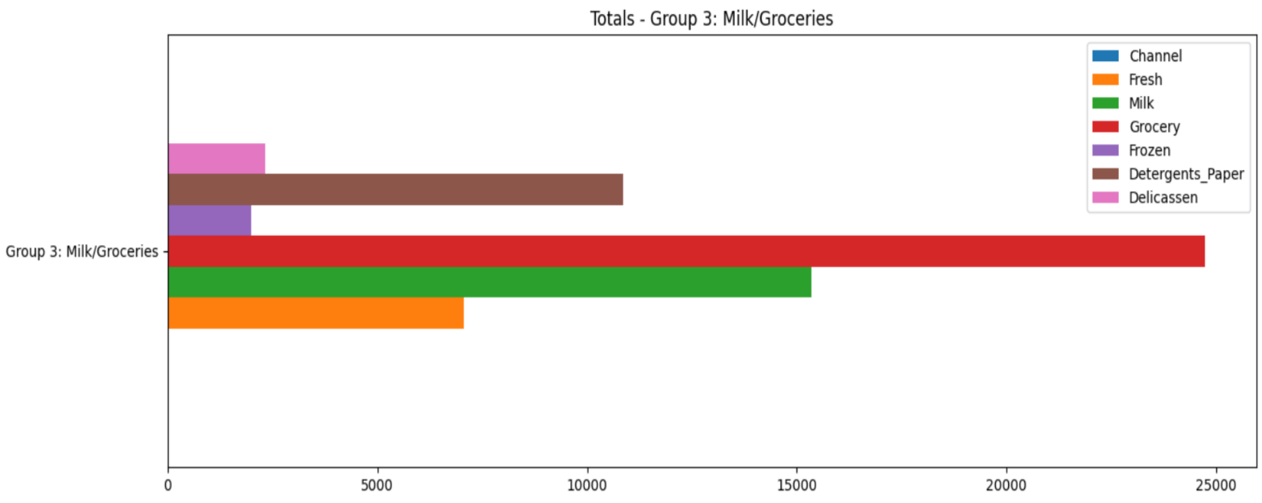
```
import matplotlib.pyplot as plt

# Create a figure with one subplot
fig, ax1 = plt.subplots(figsize=(15, 5))
ax1.set_title('Totals - Group 1:  Fresh')
# centroids
group1_data = pd.DataFrame({'Group 1:  Fresh': centroids[0]},
index=df.columns).T
group1_data.plot.barh(stacked=False, ax=ax1)
plt.show()
```



```
# Create a figure with one subplot
fig, ax2 = plt.subplots(figsize=(15, 5))
ax2.set_title('Totals - Group 2: Small Balanced')

group2_data = pd.DataFrame({'Group 2: Small Balanced': centroids[1]},
index=df.columns).T
group2_data.plot.barh(stacked=False, ax=ax2)
plt.show()
```

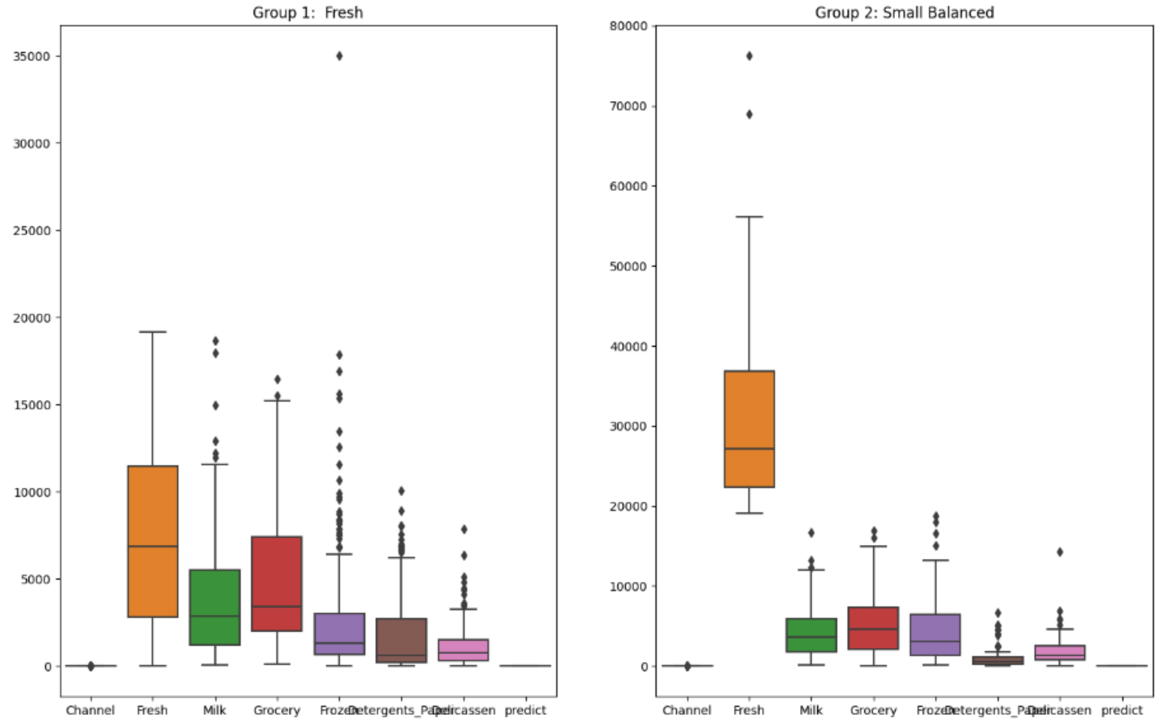


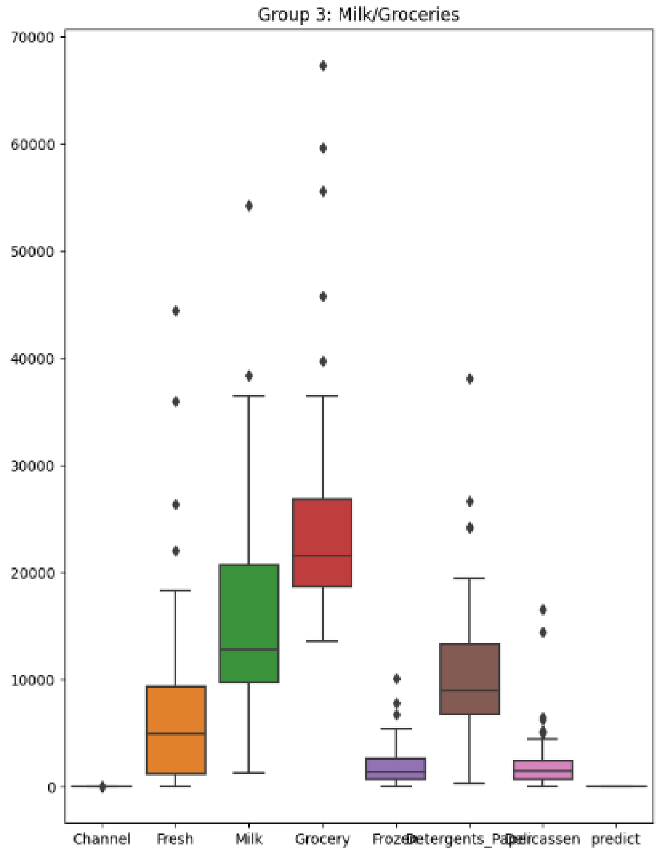
```
# Create a figure with one subplot
fig, ax3 = plt.subplots(figsize=(15, 5))
ax3.set_title('Totals - Group 3: Milk/Groceries')

group3_data = pd.DataFrame({'Group 3: Milk/Groceries': centroids[2]},
index=df.columns).T
group3_data.plot.barh(stacked=False, ax=ax3)
plt.show()
```



```
fig, (ax1, ax2, ax3) = plt.subplots(ncols=3, figsize=(25, 10))
ax1.set_title('Group 1: Fresh')
ax2.set_title('Group 2: Small Balanced')
ax3.set_title('Group 3: Milk/Groceries')
data = df.copy()
data['predict'] = predict
sns.boxplot(data=data[data['predict']==0], ax=ax1)
sns.boxplot(data=data[data['predict']==1], ax=ax2)
sns.boxplot(data=data[data['predict']==2], ax=ax3)
```

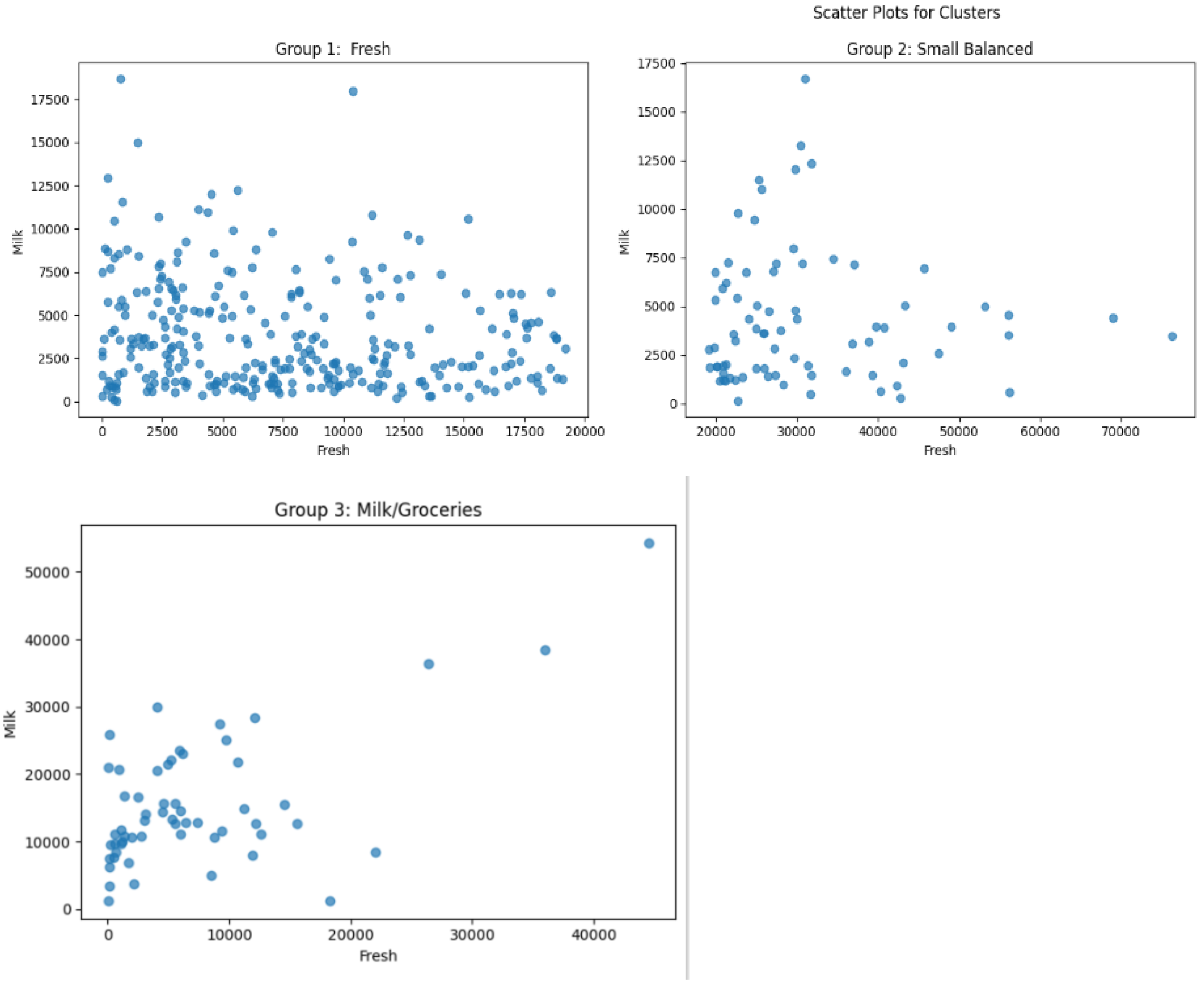
```
<Axes: title={'center': 'Group 3: Milk/Groceries'}>
```



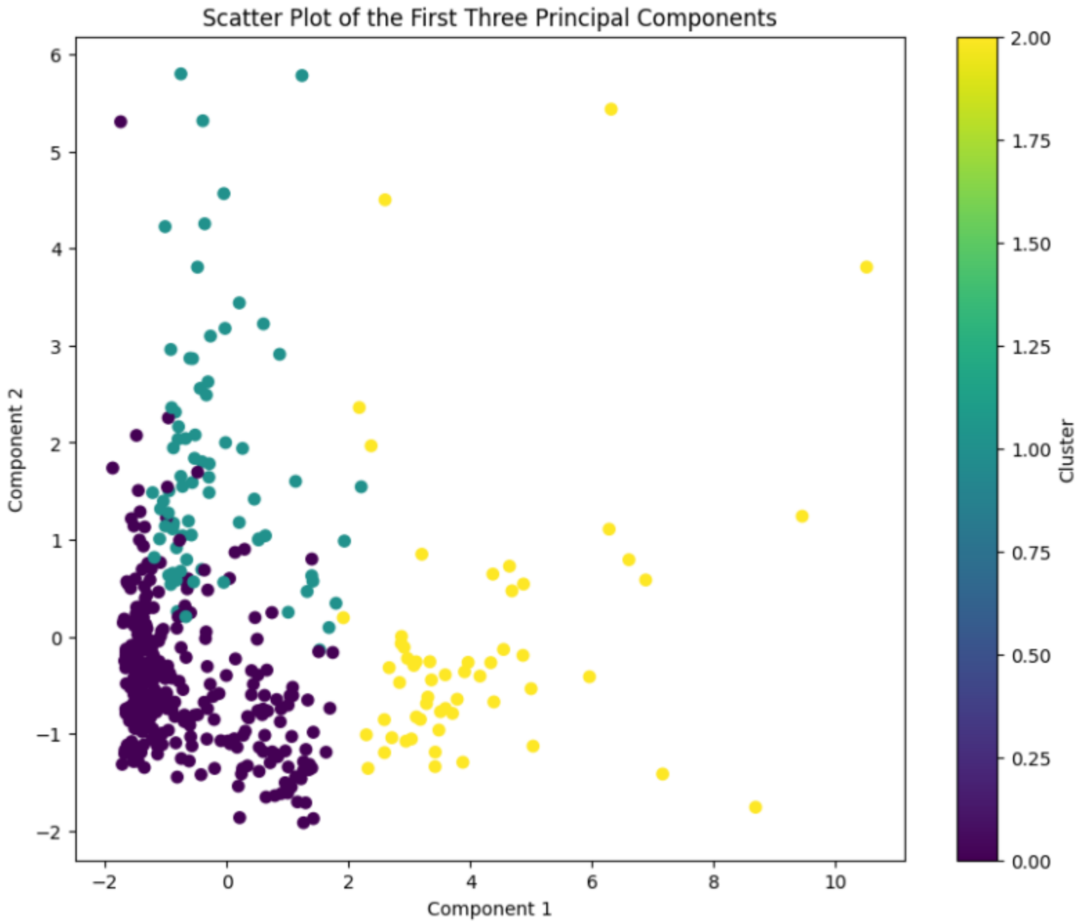


```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a figure with one subplot for each group
fig, (ax1, ax2, ax3) = plt.subplots(ncols=3, figsize=(20, 5))
fig.suptitle('Scatter Plots for Clusters')

group1_data = data[data['predict'] == 0].copy()
group2_data = data[data['predict'] == 1].copy()
group3_data = data[data['predict'] == 2].copy()

# Scatter plot for Group 1: Big Fresh
ax1.scatter(group1_data['Fresh'], group1_data['Milk'], label='Group 1: Big Fresh', alpha=0.7)
ax1.set_title('Group 1: Fresh')
ax1.set_xlabel('Fresh')
ax1.set_ylabel('Milk')

# Scatter plot for Group 2: Small Balanced
```

```
ax2.scatter(group2_data['Fresh'], group2_data['Milk'], label='Group 2:
Small Balanced', alpha=0.7)
ax2.set_title('Group 2: Small Balanced')
ax2.set_xlabel('Fresh')
ax2.set_ylabel('Milk')

# Scatter plot for Group 3: Milk/Groceries
ax3.scatter(group3_data['Fresh'], group3_data['Milk'], label='Group 3:
Milk/Groceries', alpha=0.7)
ax3.set_title('Group 3: Milk/Groceries')
ax3.set_xlabel('Fresh')
ax3.set_ylabel('Milk')

# Adjust spacing between subplots
plt.tight_layout()

plt.show()
```



```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Standardize the data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(data)

# Apply PCA to reduce dimensionality to 3 components
pca = PCA(n_components=3)
X_pca = pca.fit_transform(df_scaled)

# Create a DataFrame for the first three principal components
components_df = pd.DataFrame(data=X_pca, columns=['Component 1',
'Component 2', 'Component 3'])


# Create a scatter plot
plt.figure(figsize=(10, 8))
plt.scatter(components_df['Component 1'], components_df['Component 2'],
c=predict, cmap='viridis')
plt.title('Scatter Plot of the First Three Principal Components')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar(label='Cluster')
plt.show()
```


Scatter Plot of the First Three Principal Components
Component 1
Component 2
−2
0
2
4
6
8
10
−2
−1
0
1
2
3
4
5
6
Cluster
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Conclusion:-

1. The clustered data can be used in following ways

    a. Customer Segmentation:In businesses, clustering can be used for customer segmentation. By clustering customers based on their purchasing behavior, companies can target specific groups with personalized marketing strategies.

    b. Anomaly Detection:clustering can help identify anomalies in data. Outliers that don't belong to any cluster might indicate unusual behavior or errors in the data.

    c. Recommendation Systems:In recommendation systems, clustering can group similar items or users together. By understanding the clusters, recommendations can be made more effectively to users based on their cluster's preferences.

2. Different customers respond differently to different schemes based on their daily needs and preferences.The impact of a specific delivery scheme on different customer segments can vary significantly based on the characteristics and preferences of those segments.The delivery schemes needs to developed with accordance with unique characteristics and preference with each customer segment/