Problem description

The "Wholesale customers Data Set" is a commonly used dataset in the field of machine learning and data analysis. It contains information about various wholesale customers' annual spending habits on different product categories, making it valuable for exploring customer behavior and market segmentation. The dataset typically includes features such as spending on fresh products, milk, grocery items, frozen products, detergents, and paper products. This data can be used to analyze purchasing patterns, identify key customer segments, and develop strategies for optimizing supply chains and marketing efforts. It's a valuable resource for anyone interested in understanding wholesale customer preferences and trends.

Exploratory Data Analysis

In [1]:

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

import warnings
warnings.filterwarnings('ignore')
```

In [36]:

```
data = pd.read_csv("Wholesale customers data.csv")
data
```

Out[36]:

	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
...	...	...	...	...	...	...	...	...
435	1	3	29703	12051	16027	13135	182	2204
436	1	3	39228	1431	764	4510	93	2346
437	2	3	14531	15488	30243	437	14841	1867
438	1	3	10290	1981	2232	1038	168	2125
439	1	3	2787	1698	2510	65	477	52

440 rows × 8 columns

In [2]:
```
data = pd.read_csv("Wholesale customers data.csv")
data.drop(labels=(['Channel','Region']),axis=1,inplace=True)
data
```

Out[2]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	12669	9656	7561	214	2674	1338
1	7057	9810	9568	1762	3293	1776
2	6353	8808	7684	2405	3516	7844
3	13265	1196	4221	6404	507	1788
4	22615	5410	7198	3915	1777	5185
...	...	...	...	...	...	...
435	29703	12051	16027	13135	182	2204
436	39228	1431	764	4510	93	2346
437	14531	15488	30243	437	14841	1867
438	10290	1981	2232	1038	168	2125
439	2787	1698	2510	65	477	52

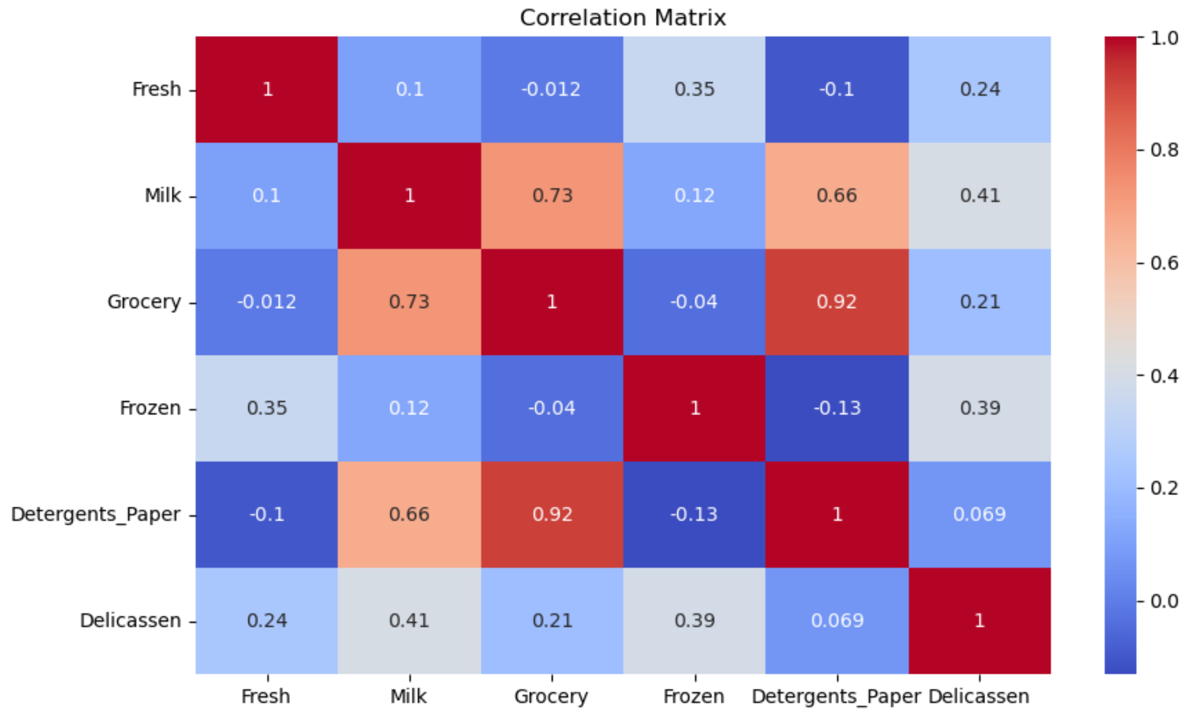
440 rows × 6 columns

In [3]:
```
data.describe()
```

Out[3]:

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

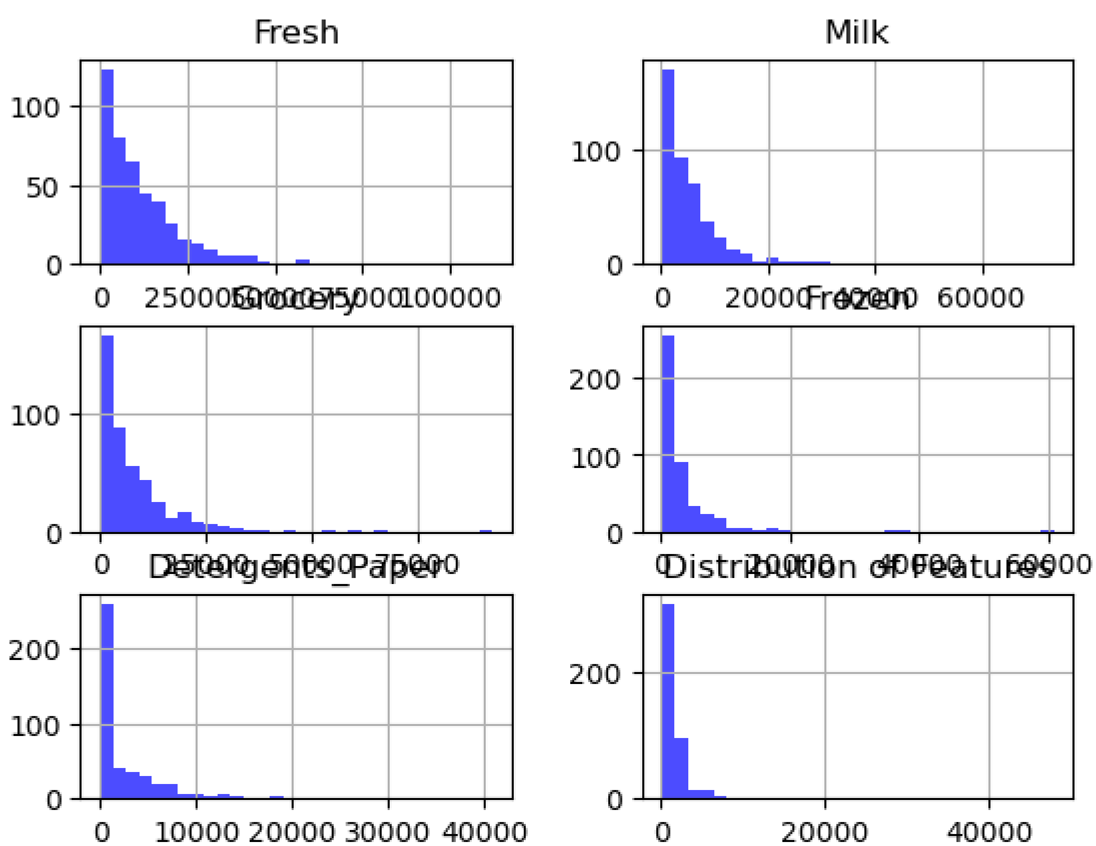
In [4]:
```
# Correlation matrix
correlation_matrix = data.corr()
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix")
plt.show()
```



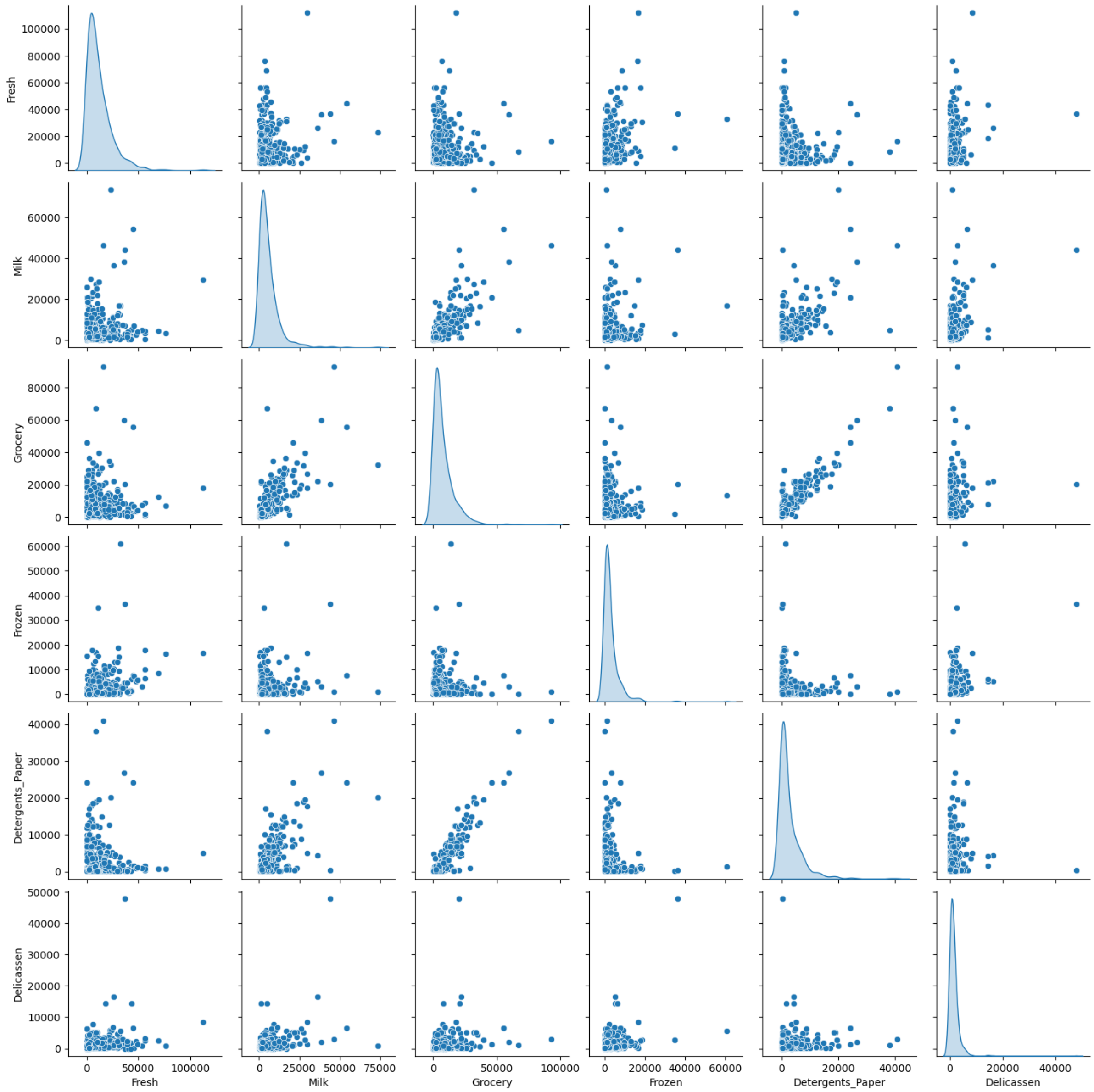
In [5]:

```
# Distribution of features
plt.figure(figsize=(12,8))
data.hist(bins=30, color='blue', alpha=0.7)
plt.title("Distribution of Features")
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



In [6]:
```python
# Pairplot to visualize relationships between features
sns.pairplot(data, diag_kind="kde")
plt.show()
```

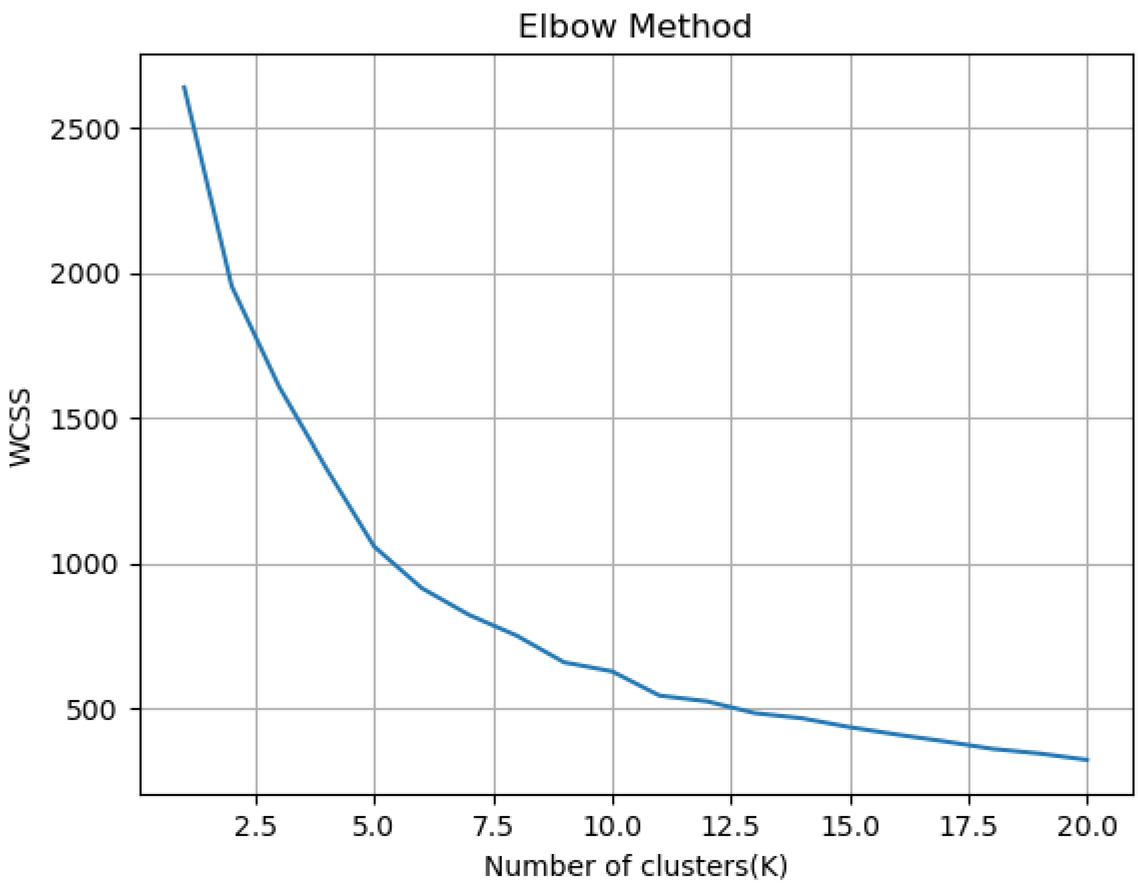


Model building and training

In [7]:
```python
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

scaler = StandardScaler()
data_scaled = pd.DataFrame(scaler.fit_transform(data), columns=data.columns)
```

In [9]:
```python
# Applying K-means Method
# Using Elbow method to fine the optimal number of K

wcss = []
max_number_of_clusters = 20
```

```
for i in range(1, max_number_of_clusters+1):
    model = KMeans(n_clusters=i, init='k-means++', random_state=42)
    model.fit(data_scaled)
    wcss.append(model.inertia_)

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



In [22]:
```
# The Elbow shape can be found in when k is 10
model = KMeans(n_clusters=5, init='k-means++', random_state=42)
model.fit(data_scaled)

# Storing the data
data_scaled['cluster'] = model.labels_
data_scaled
```

Out[22]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	cluster
0	0.052933	0.523568	-0.041115	-0.589367	-0.043569	-0.066339	1
1	-0.391302	0.544458	0.170318	-0.270136	0.086407	0.089151	1
2	-0.447029	0.408538	-0.028157	-0.137536	0.133232	2.243293	1
3	0.100111	-0.624020	-0.392977	0.687144	-0.498588	0.093411	1
4	0.840239	-0.052396	-0.079356	0.173859	-0.231918	1.299347	1
...	...	...	...	...	...	...	...
435	1.401312	0.848446	0.850760	2.075222	-0.566831	0.241091	1
436	2.155293	-0.592142	-0.757165	0.296561	-0.585519	0.291501	3
437	0.200326	1.314671	2.348386	-0.543380	2.511218	0.121456	2
438	-0.135384	-0.517536	-0.602514	-0.419441	-0.569770	0.213046	0
439	-0.729307	-0.555924	-0.573227	-0.620094	-0.504888	-0.522869	0

440 rows × 7 columns

Results and Discussion

In [26]:

```
# Description of all the clusters
for i in range(5):
    print("Cluster number : {}".format(i))
    print(data_scaled[data_scaled['cluster'] == i].describe())
    print('\n\n')
```

```
Cluster number : 0
            Fresh        Milk     Grocery      Frozen  Detergents_Paper  \
count  181.000000  181.000000  181.000000  181.000000        181.000000   
mean    -0.434037   -0.456422   -0.503317   -0.259235         -0.427114   
std      0.357575    0.267429    0.236719    0.347855          0.237584   
min     -0.949683   -0.778795   -0.823218   -0.623806         -0.604416   
25%     -0.740389   -0.651014   -0.672992   -0.517808         -0.567461   
50%     -0.458270   -0.524318   -0.584183   -0.384795         -0.530925   
75%     -0.184462   -0.344177   -0.361162   -0.116501         -0.404729   
max      0.337190    0.662201    0.383648    0.919350          0.511190   

       Delicassen  cluster  
count  181.000000    181.0  
mean    -0.294083      0.0  
std      0.190632      0.0  
min     -0.540264      0.0  
25%     -0.435894      0.0  
50%     -0.346789      0.0  
75%     -0.188104      0.0  
max      0.346526      0.0  



Cluster number : 1
           Fresh       Milk    Grocery     Frozen  Detergents_Paper  \
count  82.000000  82.000000  82.000000  82.000000         82.000000   
mean    0.121901   0.061593  -0.139444   0.602978         -0.285859   
std     0.650511   0.638539   0.444014   1.017449          0.304825   
min    -0.949683  -0.741085  -0.765698  -0.601534         -0.601897   
25%    -0.392667  -0.409458  -0.521580  -0.355976         -0.534757   
50%     0.040545  -0.045885  -0.255919   0.525776         -0.422157   
75%     0.535204   0.227888   0.142217   1.279000         -0.094436   
max     1.560499   2.405150   1.379080   3.225113          0.457016   

       Delicassen  cluster  
count   82.000000     82.0  
mean     0.372675      1.0  
std      0.748242      0.0  
min     -0.524999      1.0  
25%     -0.055068      1.0  
50%      0.207189      1.0  
75%      0.581891      1.0  
max      4.596234      1.0  



Cluster number : 2
           Fresh       Milk    Grocery     Frozen  Detergents_Paper  \
count  27.000000  27.000000  27.000000  27.000000         27.000000   
mean    0.013210   2.590663   2.748225   0.073673          2.775695   
std     0.944613   2.124159   1.798950   1.424954          1.885255   
min    -0.943192  -0.110725   1.021213  -0.626075         -0.554862   
25%    -0.609660   1.216054   1.638131  -0.442125          1.941131   
50%    -0.271932   2.118253   2.168979  -0.274673          2.277094   
75%     0.263098   3.162341   2.916844   0.035278          3.417687   
max     2.569923   9.183650   8.936528   6.900600          7.967672   
```

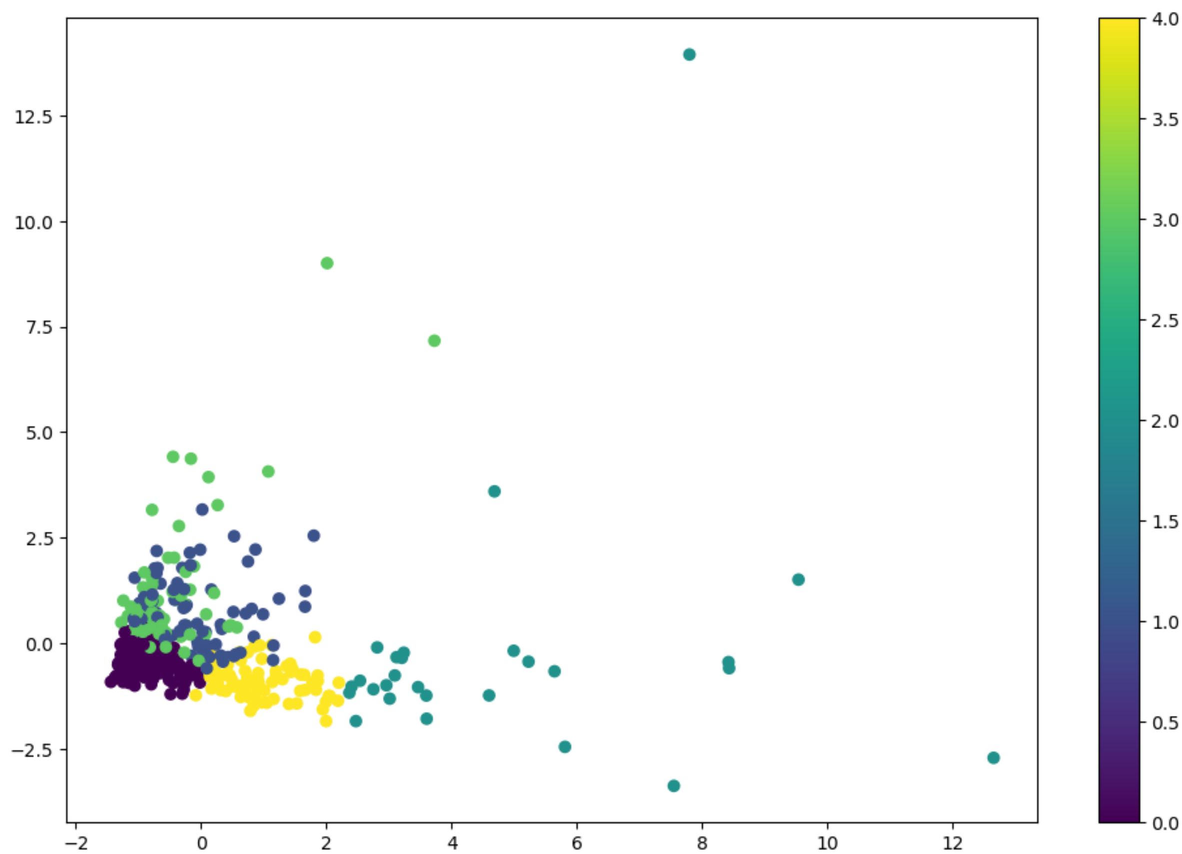
```
       Delicassen  cluster
count   27.000000     27.0
mean     1.140990      2.0
std      3.280694      0.0
min     -0.528194      2.0
25%     -0.139292      2.0
50%      0.121456      2.0
75%      1.252487      2.0
max     16.478447      2.0



Cluster number : 3
           Fresh       Milk    Grocery     Frozen  Detergents_Paper  \
count  76.000000  76.000000  76.000000  76.000000         76.000000
mean    1.477979  -0.285396  -0.343921   0.298631         -0.424392
std     1.273098   0.562410   0.399089   1.701359          0.217497
min    -0.054326  -0.768079  -0.837334  -0.609164         -0.604416
25%     0.601855  -0.599637  -0.623162  -0.400829         -0.561266
50%     1.123111  -0.395859  -0.466615  -0.110521         -0.498378
75%     2.016548  -0.192895  -0.148728   0.279032         -0.404991
max     7.927738   3.232607   1.074203  11.919002          0.433918

       Delicassen  cluster
count   76.000000     76.0
mean     0.049382      3.0
std      0.711011      0.0
min     -0.540264      3.0
25%     -0.312354      3.0
50%     -0.134144      3.0
75%      0.226536      3.0
max      4.553279      3.0



Cluster number : 4
           Fresh       Milk    Grocery     Frozen  Detergents_Paper  \
count  74.000000  74.000000  74.000000  74.000000         74.000000
mean   -0.596193   0.395999   0.736091  -0.367673          0.784568
std     0.375819   0.468764   0.488243   0.276306          0.515208
min    -0.946992  -0.613304  -0.028895  -0.628343         -0.429506
25%    -0.875235   0.065923   0.348488  -0.565703          0.428249
50%    -0.722460   0.302664   0.606406  -0.456973          0.801902
75%    -0.506081   0.713306   1.128958  -0.267713          0.984162
max     0.729576   2.015567   2.215964   0.659304          2.989755

       Delicassen  cluster
count   74.000000     74.0
mean    -0.160678      4.0
std      0.288336      0.0
min     -0.540264      4.0
25%     -0.419476      4.0
50%     -0.217037      4.0
75%      0.004306      4.0
max      0.588281      4.0
```

In [27]:
```python
from sklearn.decomposition import PCA

# Apply PCA and fit the features selected
pca = PCA(n_components=2)
pc = pca.fit_transform(data_scaled.drop('cluster', axis=1))   # Principal Component

pc_df = pd.DataFrame(pc,columns=['pc1','pc2'])
pc_df['cluster'] = data_scaled['cluster']
pc_df.head()
```

Out[27]:

	pc1	pc2	cluster
0	0.193291	-0.305100	1
1	0.434420	-0.328413	1
2	0.811143	0.815096	1
3	-0.778648	0.652754	1
4	0.166287	1.271434	1

In [35]:
```python
plt.figure(figsize=(12, 8))
scatter = plt.scatter(pc_df['pc1'], pc_df['pc2'], c=pc_df['cluster'])
plt.colorbar(scatter)  # Add a color bar
plt.show()
```



```
In [ ]:
```