Importing libraries

In [1]:

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

#libraries used for plotting
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
```

Reading the data

In [4]:

```
data = pd.read_csv("C:\\Users\\Administrator.EONEHYD016\\Downloads\\wcd.csv")
data.head()
```

Out[4]:

	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

Understanding data

In [5]:

```
data.shape
```

Out[5]:

```
(440, 8)
```

In [7]:

```
data.columns
```

Out[7]:

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

In [41]:

```
data.describe()
```

Out[41]:

	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

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
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

In [5]:

```
print('Showing Meta Data :')
data.info()
```

```
Showing Meta Data :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
Channel             440 non-null int64
Region              440 non-null int64
Fresh               440 non-null int64
Milk                440 non-null int64
Grocery             440 non-null int64
Frozen              440 non-null int64
Detergents_Paper    440 non-null int64
Delicassen          440 non-null int64
dtypes: int64(8)
memory usage: 27.6 KB
```

In [42]:

```
data.dtypes
```

Out[42]:

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

In [6]:

```
pd.isnull(data).sum()
```

Out[6]:

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

Categorical features

In [7]:

```
data.Region.value_counts()
```

Out[7]:

```
3    316
1     77
2     47
Name: Region, dtype: int64
```

In [8]:

```
data.Channel.value_counts()
```

Out[8]:

```
1    298
2    142
Name: Channel, dtype: int64
```

Detection of outliers and Clipping of data

In [10]:

```
dataset = data.copy()
```

In [11]:

```
dataset['Channel'] = dataset['Channel'].map({1:'Horeca', 2:'Retail'})
```

In [12]:

```
dataset['Region'].replace([1,2,3],['Lisbon','Oporto','other'],inplace=True)
```

In [13]:

```
dataset.head()
```

Out[13]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	Retail	other	12669	9656	7561	214	2674	1338
1	Retail	other	7057	9810	9568	1762	3293	1776
2	Retail	other	6353	8808	7684	2405	3516	7844
3	Horeca	other	13265	1196	4221	6404	507	1788
4	Retail	other	22615	5410	7198	3915	1777	5185

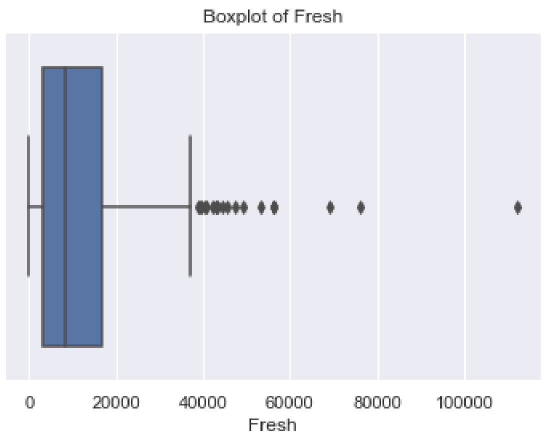
In [45]:

```
def continous_data(i):
    if dataset[i].dtype!='object':
        print('------'*10)
        sns.boxplot(dataset[i])
        plt.title("Boxplot of "+str(i))
        plt.show()
```

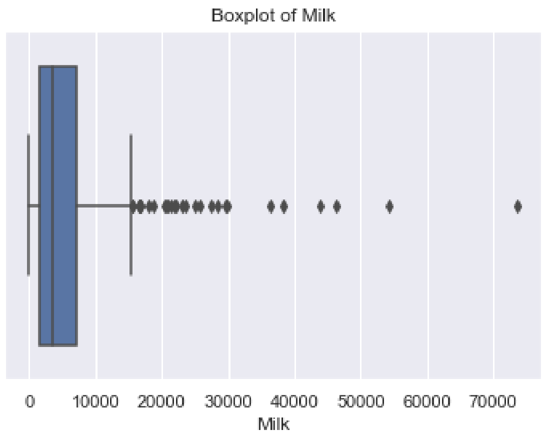
In [46]:

```
sns.set() #Sets the default seaborn style
j=['Fresh','Milk','Grocery','Frozen','Detergents_Paper','Delicassen']
for k in j:
    continous_data(i=k)
```

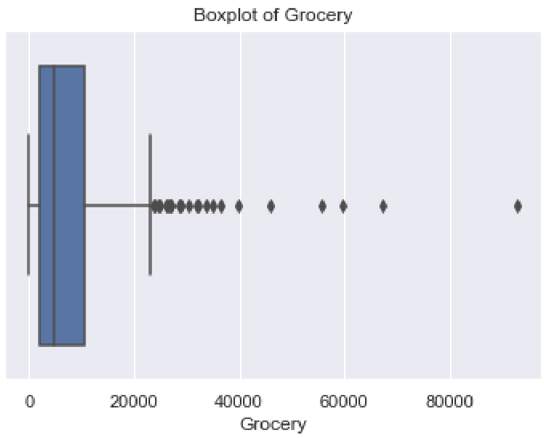
```
------------------------------------------------------------
```


Boxplot of Fresh
0
20000
40000
60000
80000
100000
Fresh

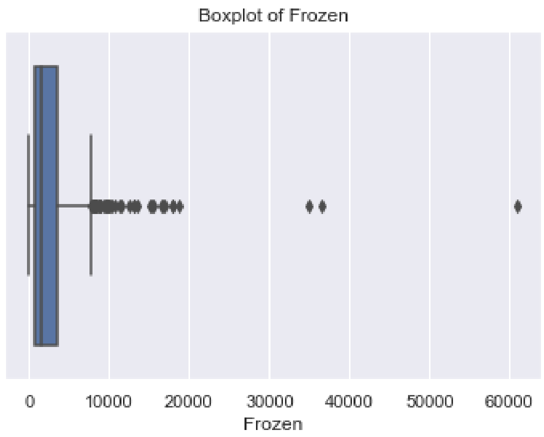
```
-----------------------------------------------------------------
```


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

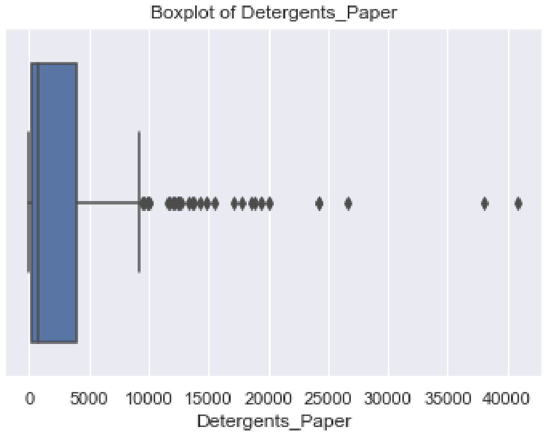
```
-----------------------------------------------------------------
```


Boxplot of Grocery
0
20000
40000
60000
80000
Grocery

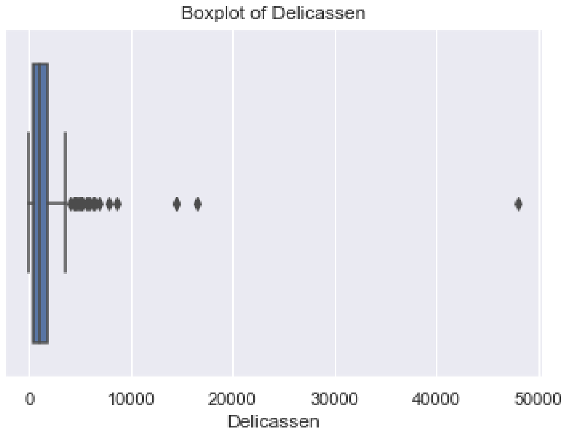
```
-----------------------------------------------------------------
```


Boxplot of Frozen
0
10000
20000
30000
40000
50000
60000
Frozen

```
------------------------------------------------------------
```



```
------------------------------------------------------------
```



In [16]:

```python
bf=data[['Fresh','Milk','Grocery','Frozen','Detergents_Paper','Delicassen']]
```

In [17]:

```python
def quantile(s):
    data = bf[s]
    print(data.quantile(0.9))
    print(data.quantile(0.95))
    print(data.quantile(0.99))
    print(data.quantile(1))
```

In [18]:

```python
for x in bf:
    print(x)
    quantile(x)
    print('\n\n')
```

```
Fresh
27090.500000000004
36818.5
56082.61
112151.0



Milk
12229.900000000001
16843.399999999947
37610.06000000003
73498.0
```

```
Grocery
18910.10000000001
24033.499999999967
43435.74000000008
92780.0



Frozen
7545.300000000004
9930.749999999996
17964.82
60869.0



Detergents_Paper
7438.300000000003
12043.199999999992
22571.61000000006
40827.0



Delicassen
2945.9000000000005
4485.399999999994
8274.660000000009
47943.0
```

In [19]:

```
log_data =
np.log(dataset[['Fresh','Milk','Grocery','Frozen','Detergents_Paper','Delicassen']].copy())
```

In [48]:

```
log_data.head()
```

Out[48]:

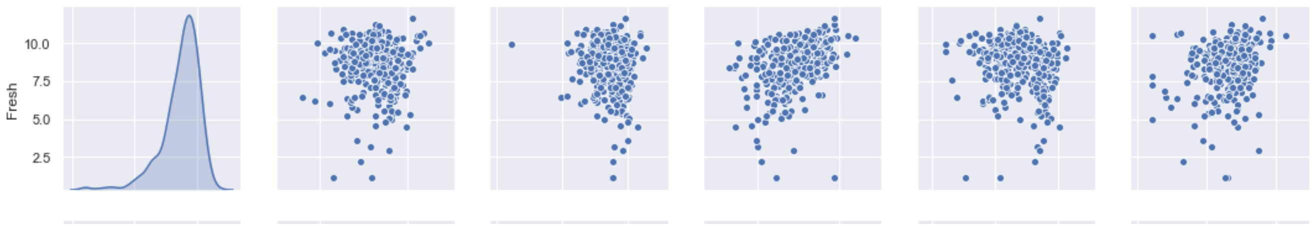
	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931
1	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119
2	8.756682	9.083416	8.946896	7.785305	8.165079	7.988168
3	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853
4	10.026369	8.596004	8.881558	8.272571	7.482682	7.988168

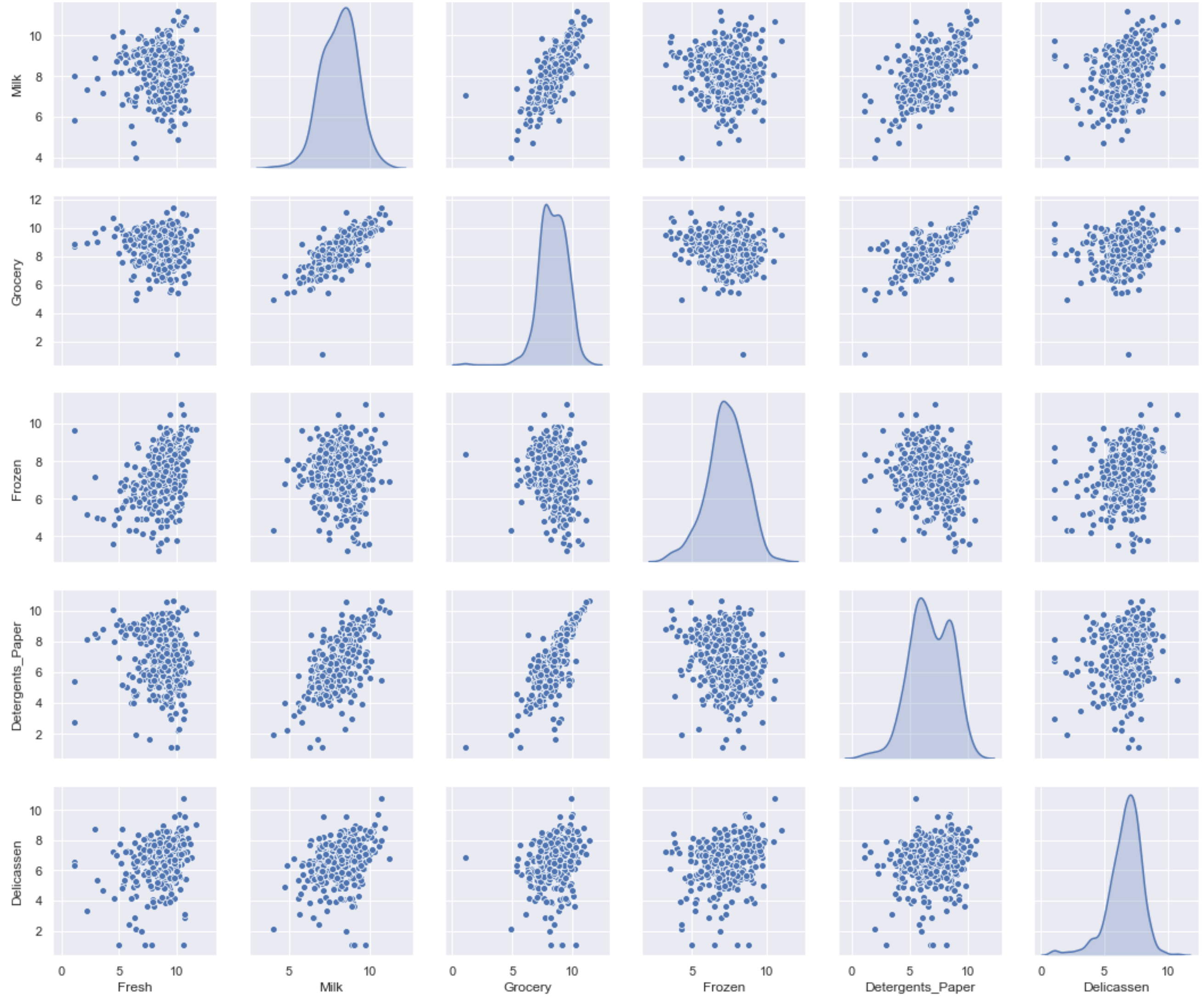
In [21]:

```
sns.pairplot(log_data,diag_kind = 'kde')
```

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x2aa237345f8>
```





In [22]:

```
log_data['Milk']=log_data['Milk'].clip(0,log_data['Milk'].quantile(0.90))
log_data['Fresh']=log_data['Fresh'].clip(0,log_data['Fresh'].quantile(0.95))
log_data['Grocery']=log_data['Grocery'].clip(0,log_data['Grocery'].quantile(0.90))
log_data['Frozen']=log_data['Frozen'].clip(0,log_data['Frozen'].quantile(0.90))
log_data['Detergents_Paper']=log_data['Detergents_Paper'].clip(0,log_data['Detergents_Paper'].quant
ile(0.90))
log_data['Delicassen']=log_data['Delicassen'].clip(0,log_data['Delicassen'].quantile(0.90))
```

Dummification

In [23]:

```
df  =  pd.concat([dataset[['Channel','Region']],log_data],axis=1)
df.head()
```

Out[23]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	Retail	other	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931
1	Retail	other	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119
2	Retail	other	8.756682	9.083416	8.946896	7.785305	8.165079	7.988168
3	Horeca	other	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853
4	Retail	other	10.026369	8.596004	8.881558	8.272571	7.482682	7.988168

In [24]:

```
df = pd.get_dummies(df,columns=['Channel','Region'],drop_first=True)
df.head()
```

Out[24]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Channel_Retail	Region_Oporto	Region_other
0	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931	1	0	1
1	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119	1	0	1
2	8.756682	9.083416	8.946896	7.785305	8.165079	7.988168	1	0	1
3	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853	0	0	1
4	10.026369	8.596004	8.881558	8.272571	7.482682	7.988168	1	0	1

In [72]:

```
import pandas_profiling #importing pandas profiling
```

In [73]:

```
profile = pandas_profiling.ProfileReport(df)
rejected_variables = profile.get_rejected_variables(threshold=0.9)
```

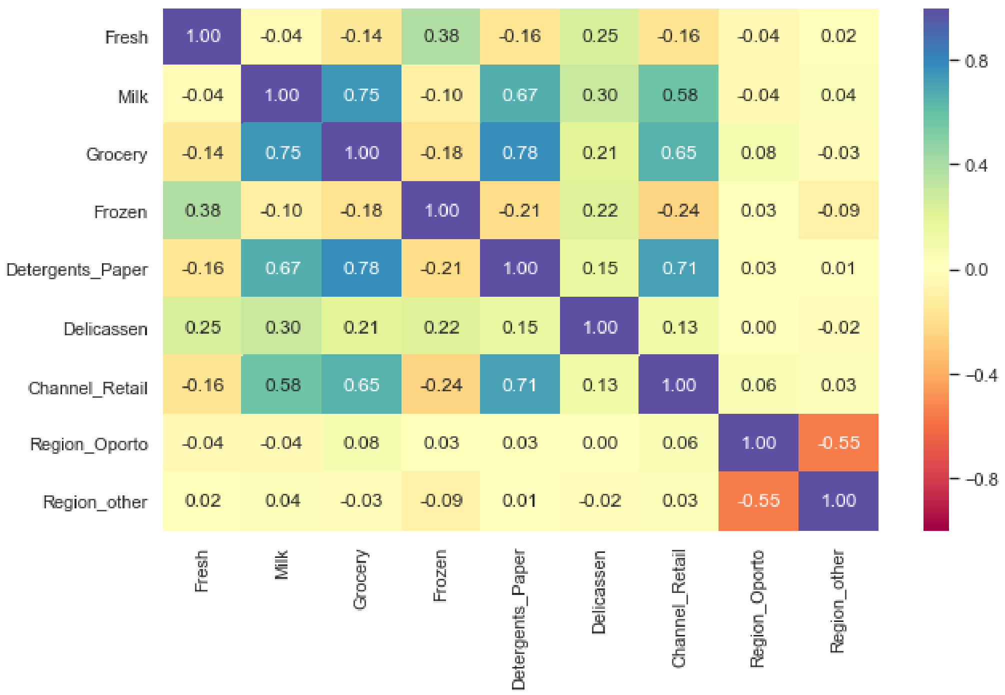
In [75]:

```
profile.to_file(outputfile="C:\\Users\\c82020\\Desktop\\telco\\profile_new.html")
```

In [25]:

```
print('Correlation Heat map of the data')
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True,fmt='.2f',vmin=-1,vmax=1,cmap='Spectral')
plt.show()
```

```
Correlation Heat map of the data
```

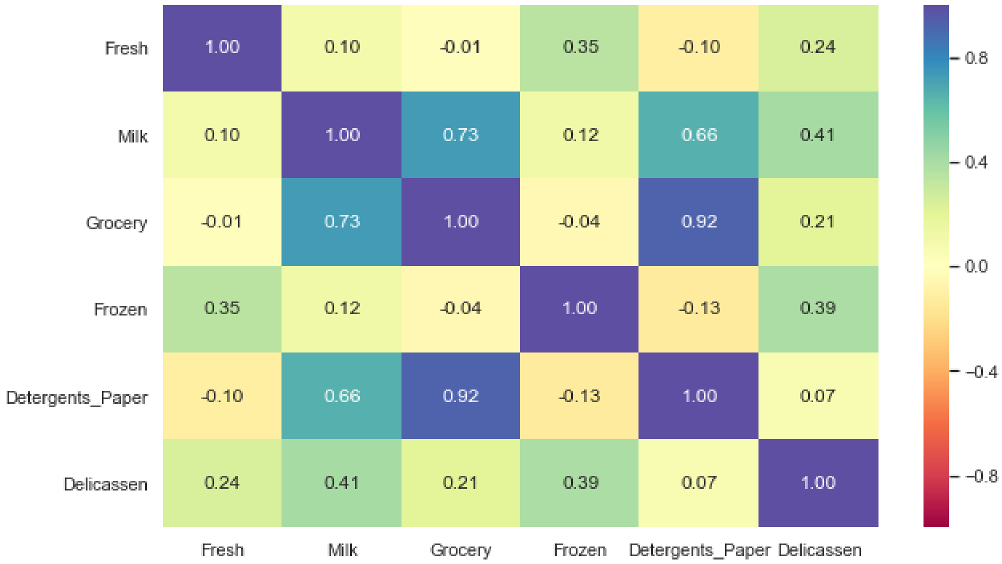


Correlation

In [71]:

```
print('Correlation Heat map of the data')
print('Correlation Heat map of the data')
plt.figure(figsize=(10,6))
sns.heatmap(dataset.corr(),annot=True,fmt='.2f',vmin=-1,vmax=1,cmap='Spectral')
plt.show()
```

```
Correlation Heat map of the data
```

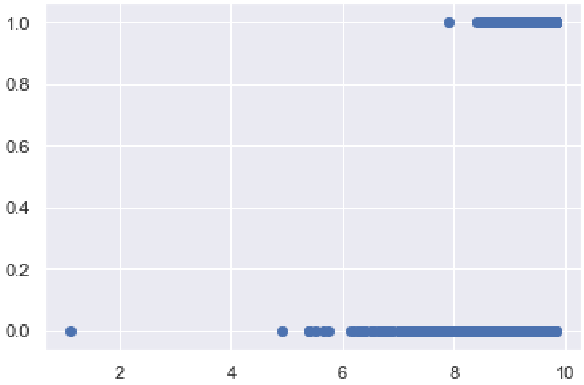


In [62]:

```
def scatterplot(i,j):
    plt.scatter(data=df,x=i,y=j)
    plt.show()
```

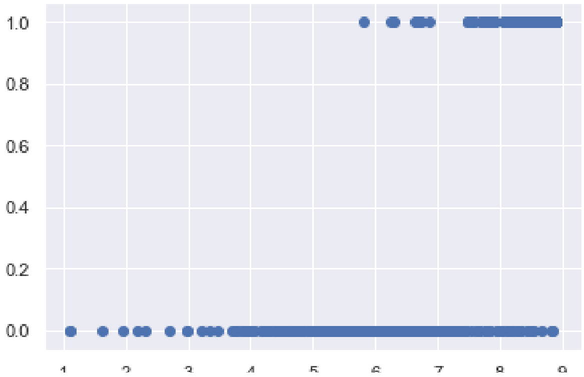
In [63]:

```
scatterplot(i='Grocery',j='Channel_Retail')
```



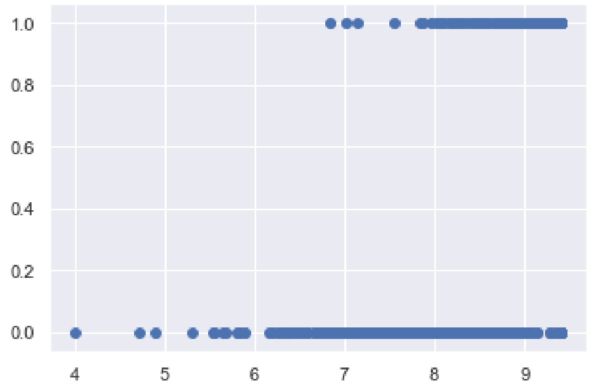
In [64]:

```
scatterplot(i='Detergents_Paper',j='Channel_Retail')
```



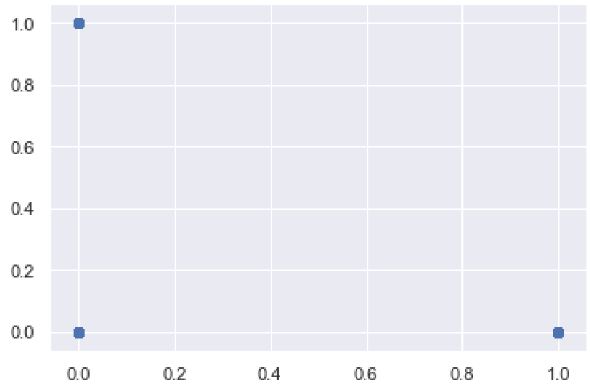
In [65]:

```
scatterplot(i='Milk',j='Channel_Retail')
```



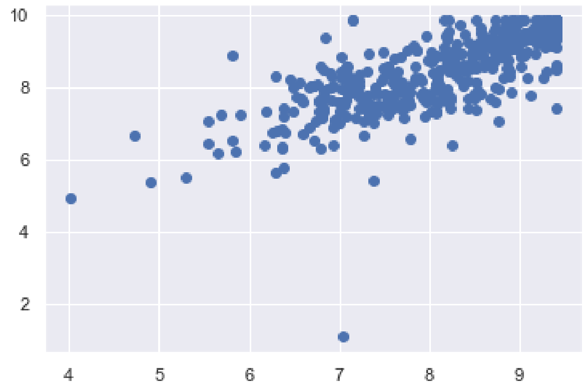
In [66]:

```
scatterplot(i='Region_Oporto',j='Region_other')
```



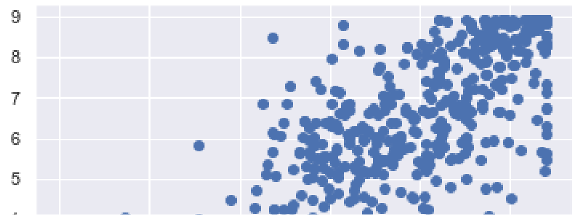
In [67]:

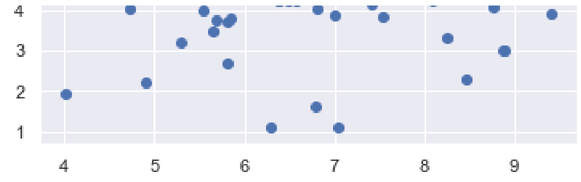
```
scatterplot(i='Milk',j='Grocery')
```



In [68]:

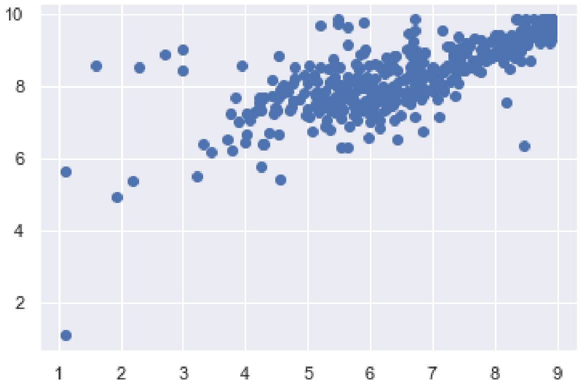
```
scatterplot(i='Milk',j='Detergents_Paper')
```





In [69]:

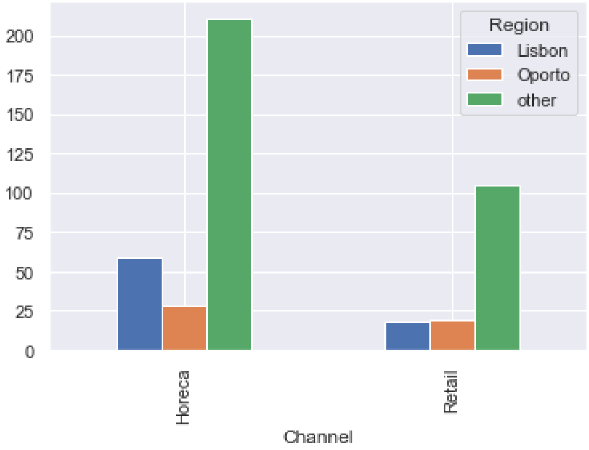
```
scatterplot(i='Detergents_Paper',j='Grocery')
```



In [44]:

```
def categorical_multi(i,j):
    pd.crosstab(dataset[i],dataset[j]).plot(kind='bar')
    plt.show()
    print(pd.crosstab(dataset[i],dataset[j]))

categorical_multi(i='Channel',j='Region')
```



```
Region   Lisbon  Oporto  other
Channel
Horeca       59      28    211
Retail       18      19    105
```

kmean

In [26]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_std = scaler.fit_transform(df)
df_std = pd.DataFrame(df_std,columns=df.columns)
df_std.head()
```

```
D:\Anaconda\lib\site-packages\sklearn\preprocessing\data.py:323: DataConversionWarning: Data with
```

```
input dtype uint8, float64 were all converted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

Out[26]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Channel_Retail	Region_Oporto	Region_other
0	0.886689	0.956275	0.895222	0.376039	0.869104	0.885444	1.0	0.0	1.0
1	0.824540	0.959203	0.922131	0.745269	0.895746	0.926548	1.0	0.0	1.0
2	0.813378	0.939267	0.897067	0.799755	0.904129	1.000000	1.0	0.0	1.0
3	0.891571	0.569806	0.828593	0.971280	0.656352	0.927526	0.0	0.0	1.0
4	0.948234	0.849077	0.889599	0.885094	0.816819	1.000000	1.0	0.0	1.0

In [27]:

```
kf = df_std.copy()
```

In [28]:

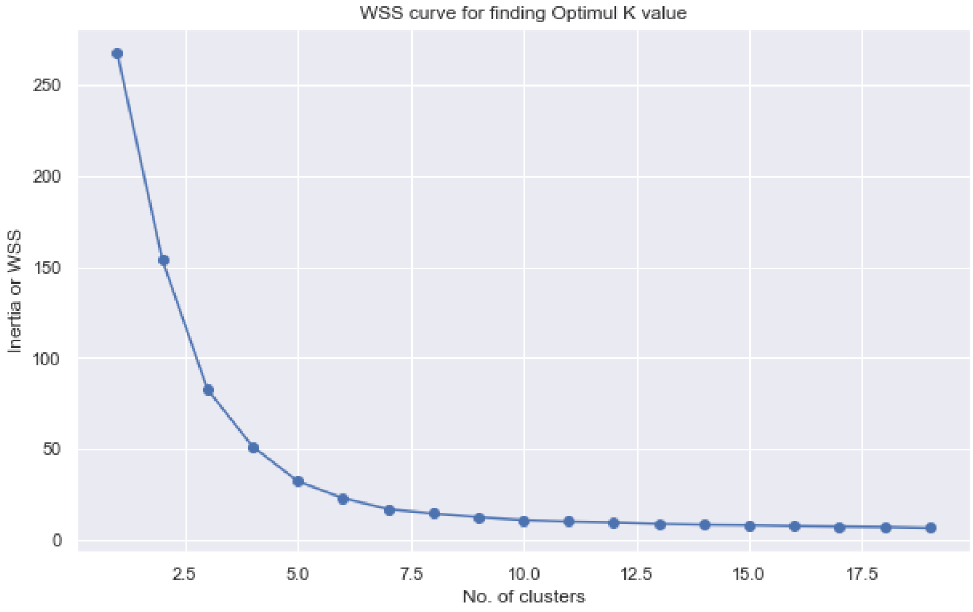
```
X = kf[['Milk','Grocery','Detergents_Paper','Channel_Retail','Region_Oporto','Region_other']]
```

In [29]:

```
from sklearn.cluster import KMeans
cluster_range = range(1,20)
cluster_wss=[]
for cluster in cluster_range:
    model = KMeans(cluster)
    model.fit(X)
    cluster_wss.append(model.inertia_)
```

In [30]:

```
plt.figure(figsize=[10,6])
plt.title('WSS curve for finding Optimul K value')
plt.xlabel('No. of clusters')
plt.ylabel('Inertia or WSS')
plt.plot(list(cluster_range),cluster_wss,marker='o')
plt.show()
```



In [31]:

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=6,random_state=100,n_jobs=1)
model.fit(X)
```

Out[31]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=6, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=100, tol=0.0001, verbose=0)
```

In [32]:

```
from sklearn import metrics
metrics.silhouette_score(X, model.labels_)
```

Out[32]:

```
0.7458207619142911
```

In [40]:

```
from sklearn.metrics import confusion_matrix,classification_report
print(confusion_matrix(dataset_final['clusters'],model.labels_))
print(classification_report(dataset_final['clusters'],model.labels_))
```

```
[[211   0   0   0   0   0]
 [  0  59   0   0   0   0]
 [  0   0 105   0   0   0]
 [  0   0   0  19   0   0]
 [  0   0   0   0  28   0]
 [  0   0   0   0   0  18]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       211
           1       1.00      1.00      1.00        59
           2       1.00      1.00      1.00       105
           3       1.00      1.00      1.00        19
           4       1.00      1.00      1.00        28
           5       1.00      1.00      1.00        18

   micro avg       1.00      1.00      1.00       440
   macro avg       1.00      1.00      1.00       440
weighted avg       1.00      1.00      1.00       440
```

In [33]:

```
model.labels_
```

Out[33]:

```
array([2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 0,
       0, 2, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 0, 0, 0, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 0, 0, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2,
       0, 2, 0, 0, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 2, 0, 2, 2, 2, 0,
       0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 0, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2,
       0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 2, 2, 0, 2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 0, 0, 2, 2, 0, 2, 0, 2,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 2, 0, 0, 1, 5,
       1, 1, 5, 5, 1, 1, 1, 5, 1, 5, 1, 5, 1, 5, 1, 1, 5, 1, 5, 1, 5, 1,
       1, 1, 1, 5, 1, 1, 5, 1, 1, 1, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 5, 1, 1, 1, 1, 1, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       5, 1, 5, 1, 5, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 3, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4,
       4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 3, 4, 3, 3, 3, 4, 4, 4, 4, 2, 2, 0, 2, 0, 0, 2, 2, 0, 2, 0, 2,
       0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2,
       0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 2, 2, 0,
       2, 0, 0, 2, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0])
```

In [34]:

```
model.cluster_centers_
```

Out[34]:

```
array([[ 6.81405925e-01,  7.76230272e-01,  6.15390419e-01,
         4.99600361e-16,  2.08166817e-16,  1.00000000e+00],
       [ 6.90491157e-01,  7.84410806e-01,  6.34490081e-01,
         5.55111512e-17,  8.32667268e-17, -3.33066907e-16],
       [ 9.08432714e-01,  9.47494108e-01,  9.35807305e-01,
         1.00000000e+00, -6.93889390e-17,  1.00000000e+00],
       [ 8.75963102e-01,  9.42777927e-01,  9.40020315e-01,
         1.00000000e+00,  1.00000000e+00, -1.11022302e-16],
       [ 6.31468584e-01,  8.10622810e-01,  5.99551244e-01,
         2.77555756e-16,  1.00000000e+00, -2.22044605e-16],
       [ 9.25689425e-01,  9.59990896e-01,  9.45295030e-01,
         1.00000000e+00, -4.16333634e-17, -1.11022302e-16]])
```

In [35]:

```
dataset_final = data.copy()
dataset_final.head()
```

Out[35]:

	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

In [36]:

```
dataset_final['clusters']=model.predict(X)
dataset_final.head()
```

Out[36]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	clusters
0	2	3	12669	9656	7561	214	2674	1338	2
1	2	3	7057	9810	9568	1762	3293	1776	2
2	2	3	6353	8808	7684	2405	3516	7844	2
3	1	3	13265	1196	4221	6404	507	1788	0
4	2	3	22615	5410	7198	3915	1777	5185	2

In [37]:

```
clust_prof = dataset_final.groupby(['clusters'],as_index=False).mean()
clust_prof
```

Out[37]:

	clusters	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	0	1.0	3.0	13878.052133	3486.981043	3886.734597	3656.900474	786.682464	1518.284360
1	1	1.0	1.0	12902.254237	3870.203390	4026.135593	3127.322034	950.525424	1197.152542
2	2	2.0	3.0	9831.504762	10981.009524	15953.809524	1513.200000	6899.238095	1826.209524
3	3	2.0	2.0	7289.789474	9190.789474	16326.315789	1540.578947	8410.263158	1239.000000
4	4	1.0	2.0	11650.535714	2304.250000	4395.500000	5745.035714	482.714286	1105.892857
5	5	2.0	1.0	5200.000000	10784.000000	18471.944444	2584.111111	8225.277778	1871.944444

PCA

In [119]:

```
from sklearn.decomposition import PCA
pca2 = PCA(n_components=2)
pc = pca2.fit_transform(df_std)
pc_df = pd.DataFrame(pc)
pc_df.head()
```

Out[119]:

	0	1
0	0.777331	-0.227159
1	0.753272	-0.213701
2	0.745141	-0.211639
3	-0.344032	-0.326709
4	0.676513	-0.216450

In [221]:

```
from sklearn.cluster import KMeans
Kmean = KMeans(n_clusters=6)
Kmean.fit(pc)
```

Out[221]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=6, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [223]:

```
Kmean.cluster_centers_
```

Out[223]:

```
array([[-0.31545602, -0.33630991],
       [ 0.77020429, -0.21592524],
       [-0.4293831 ,  0.54283316],
       [ 0.65435898,  0.66634028],
       [-0.45713622,  1.00776838],
       [ 0.63393289,  1.12603917]])
```

In [123]:

```
pca = pd.concat([pc_df,dataset_final['clusters']],axis=1)
pca.columns = ['pc1','pc2','clusters']
print(pca.shape)
pca.head()
```

```
(440, 3)
```

Out[123]:

	pc1	pc2	clusters
0	0.777331	-0.227159	2
1	0.753272	-0.213701	2
2	0.745141	-0.211639	2
3	-0.344032	-0.326709	0
4	0.676513	-0.216450	2

In [124]:

```
pca.clusters.value_counts()
```

Out[124]:

```
0    211
2    105
1     59
4     28
3     19
5     18
Name: clusters, dtype: int64
```

In [ ]:

```
from sklearn.cluster import KMeans
Kmean = KMeans(n_clusters=6)
Kmean.fit(pc)
```

In [ ]:

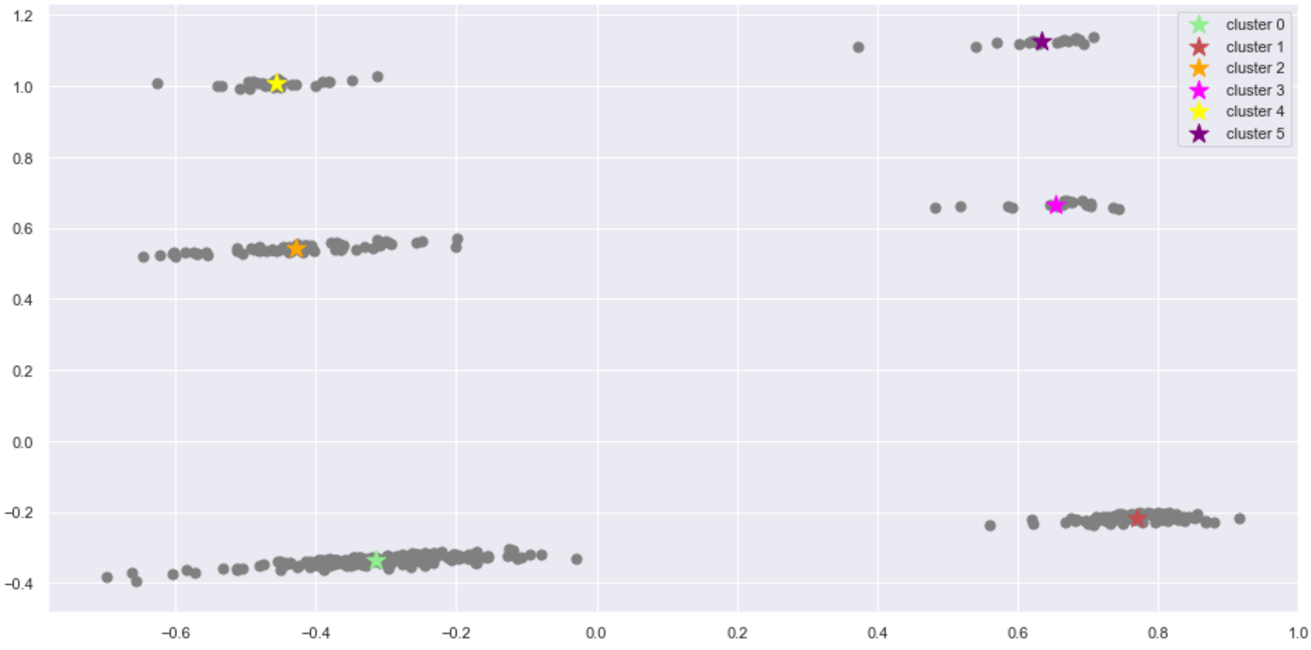
```
Kmean.cluster_centers_
```

In [ ]:

```
pca = pd.concat([pc_df,dataset_final['clusters']],axis=1)
pca.columns = ['pc1','pc2','clusters']
print(pca.shape)
pca.head()
```

In [261]:

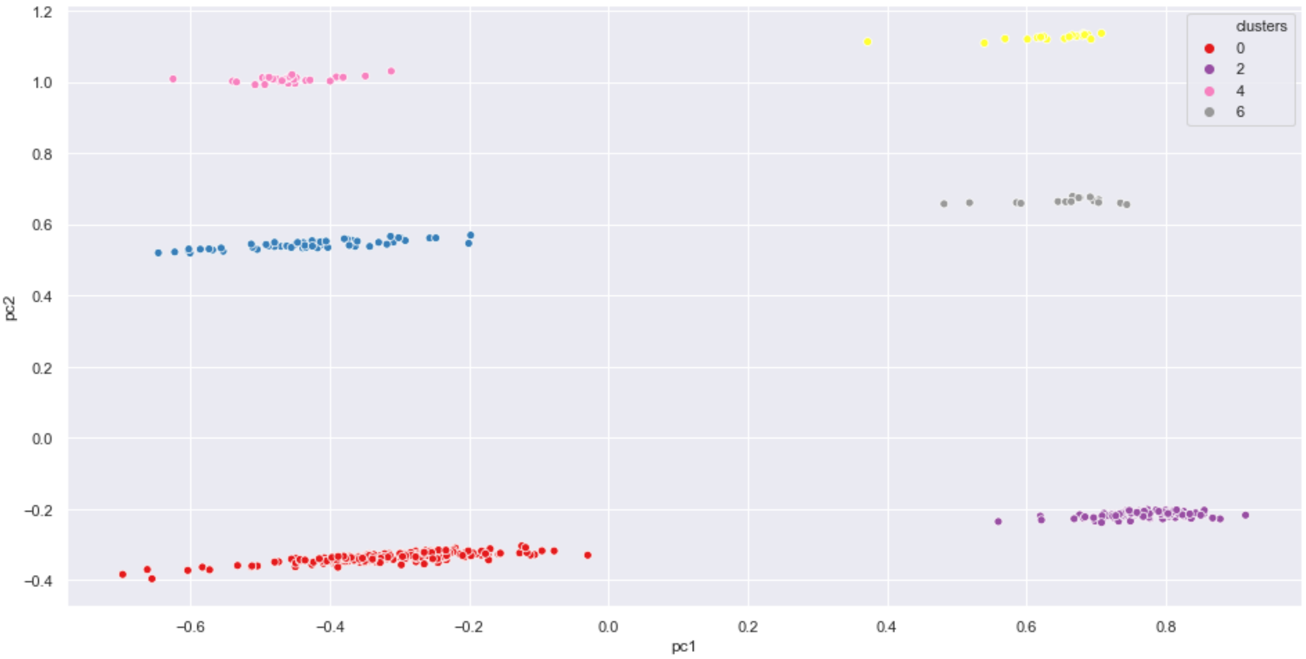
```
plt.figure(figsize=[16,8])
plt.scatter(X[ : , 0], X[ : , 1], s =50, c='grey')
plt.scatter( -0.31545602, -0.33630991,s=200, c='lightgreen', marker='*', label='cluster 0')
plt.scatter( 0.77020429, -0.21592524,s=200, c='r', marker='*',label='cluster 1')
plt.scatter( -0.4293831 ,  0.54283316,s=200, c='orange', marker='*',label='cluster 2')
plt.scatter(0.65435898,  0.66634028,s=200, c='magenta', marker='*',label='cluster 3')
plt.scatter( -0.45713622,  1.00776838,s=200, c='yellow', marker='*', label='cluster 4')
plt.scatter( 0.63393289,  1.12603917,s=200, c='purple', marker='*', label='cluster 5')
plt.legend()
plt.show()
```



In [256]:

```
plt.figure(figsize=[16,8])
cluster=[0,1,2,3,4,5]
```

```
sns.scatterplot(x='pc1', y='pc2',hue='clusters',data=pca,palette='Set1')
plt.legend()
plt.show()
```



In [ ]:

```

```