

# Chapter 11 - Exercise 4: Online Retail

- Cho dữ liệu Online\_Retail.xlsx
- Hãy thực hiện việc phân cụm khách hàng dựa trên các thông tin 'TotalSales', 'OrderCount', 'AvgOrderValue' thu thập và tính toán từ dataset trên. ## Yêu cầu:
- Đọc dữ liệu. Chuẩn hóa dữ liệu. Trích xuất các thuộc tính cần thiết.
- Tìm số cụm phù hợp k?
- Áp dụng thuật toán K-Means để giải bài toán phân cụm với số cụm đã tìm được ở câu 2.
- Vẽ hình, xem kết quả. Giải thích từng cụm

## Gợi ý:

In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

## 1. Đọc dữ liệu, chuẩn hóa dữ liệu. Trích xuất các thuộc tính cần thiết.

In [2]:

```
df = pd.read_excel('Online_Retail.xlsx', sheet_name='Online Retail')
```

In [59]:

```
df.shape
```

Out[59]:

```
(380620, 9)
```

In [60]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 380620 entries, 0 to 516368
Data columns (total 9 columns):
InvoiceNo      380620 non-null object
StockCode      380620 non-null object
Description    380620 non-null object
Quantity       380620 non-null int64
InvoiceDate    380620 non-null datetime64[ns]
UnitPrice      380620 non-null float64
CustomerID     380620 non-null float64
Country        380620 non-null object
Sales          380620 non-null float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(4)
memory usage: 29.0+ MB
```

In [61]:

```
df.head()
```

Out[61]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T- LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Unitec Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Unitec Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom

In [62]:

```
# Loại bỏ các dòng có Quantity âm (Là hàng bị hủy)
df.loc[df['Quantity'] <= 0].shape
```

Out[62]:

(0, 9)

In [63]:

```
df = df.loc[df['Quantity'] > 0]
```

In [64]:

```
df.shape
```

Out[64]:

```
(380620, 9)
```

In [65]:

```
# Loại bỏ các dòng có CustomerID = NULL  
df = df[pd.notnull(df['CustomerID'])]  
df.shape
```

Out[65]:

```
(380620, 9)
```

In [66]:

```
df.head()
```

Out[66]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T- LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Unitec Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Unitec Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom

In [67]:

```
# kiểm tra dữ liệu null
print(df.isnull().sum())
# => Không còn dữ liệu null
```

```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
Sales          0
dtype: int64
```

In [68]:

```
# Loại bỏ dữ liệu trong tháng chưa hoàn chỉnh là tháng 12/2011
print('Date Range: %s ~ %s' % (df['InvoiceDate'].min(), df['InvoiceDate'].max()))
```

Date Range: 2010-12-01 08:26:00 ~ 2011-11-30 17:37:00

In [69]:

```
df.loc[df['InvoiceDate'] >= '2011-12-01'].shape
```

Out[69]:

(0, 9)

In [70]:

```
df = df.loc[df['InvoiceDate'] < '2011-12-01']
```

In [71]:

```
df.shape
```

Out[71]:

(380620, 9)

In [72]:

```
# Tính total sales (Sales = Quantity * UnitPrice)
df['Sales'] = df['Quantity'] * df['UnitPrice']
df.head()
```

Out[72]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T- LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Unitec Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Unitec Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Unitec Kingdom

In [73]:

```
# Tính TotalSales, OrderCount, AvgOrderValue cho từng khách hàng CustomerID => customer_df
customer_df = df.groupby('CustomerID').agg({
    'Sales': sum,
    'InvoiceNo': lambda x: x.nunique()
})
```

In [74]:

```
customer_df.head()
```

Out[74]:

	Sales	InvoiceNo
CustomerID		
12346.0	77183.60	1
12347.0	4085.18	6
12348.0	1797.24	4
12349.0	1757.55	1
12350.0	334.40	1

In [75]:

```
# # Tạo các cột thuộc tính TotalSales, OrderCount, AvgOrderValue cho customer_df
customer_df.columns = ['TotalSales', 'OrderCount']
customer_df['AvgOrderValue'] = customer_df['TotalSales']/customer_df['OrderCount']
customer_df.head()
```

Out[75]:

	TotalSales	OrderCount	AvgOrderValue
CustomerID			
12346.0	77183.60	1	77183.600000
12347.0	4085.18	6	680.863333
12348.0	1797.24	4	449.310000
12349.0	1757.55	1	1757.550000
12350.0	334.40	1	334.400000

In [76]:

```
# Thống kê chung
customer_df.describe()
```

Out[76]:

	TotalSales	OrderCount	AvgOrderValue
count	4298.000000	4298.000000	4298.000000
mean	1952.818779	4.131689	400.255621
std	8354.913254	7.420253	1271.187289
min	0.000000	1.000000	0.000000
25%	304.305000	1.000000	178.602500
50%	657.265000	2.000000	295.033958
75%	1599.515000	4.000000	431.594250
max	268478.000000	201.000000	77183.600000

In [77]:

```
import numpy as np
np.ptp(customer_df.TotalSales)
```

```
c:\program files\python36\lib\site-packages\numpy\core\fromnumeric.py:254
2: FutureWarning: Method .ptp is deprecated and will be removed in a futur
e version. Use numpy.ptp instead.
    return ptp(axis=axis, out=out, **kwargs)
```

Out[77]:

268477.9999999998

In [78]:

```
np.ptp(customer_df.OrderCount)
```

Out[78]:

200

In [79]:

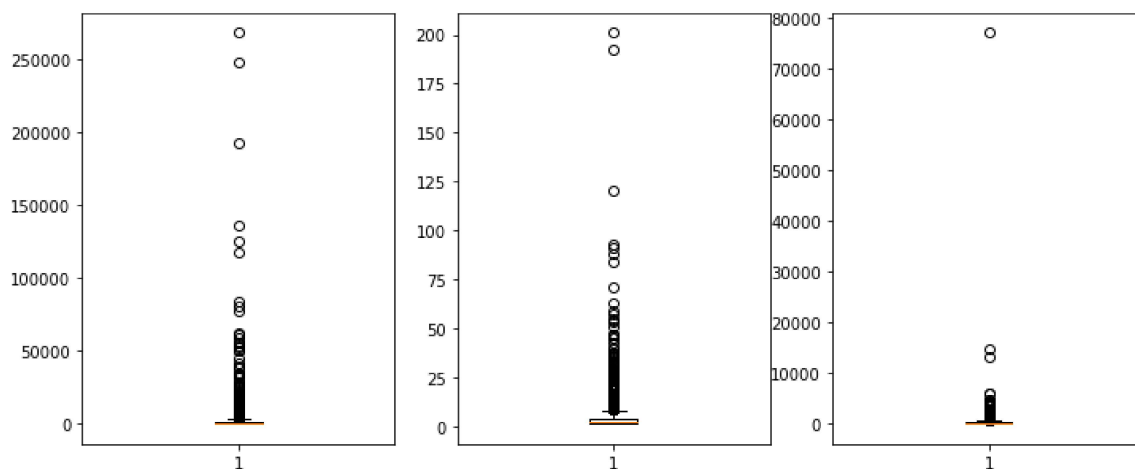
```
np.ptp(customer_df.AvgOrderValue)
```

Out[79]:

77183.6

In [80]:

```
# => Có sự khác biệt về thang đo giữa các cột dữ liệu
plt.figure(figsize=(12,5))
plt.subplot(1,3,1)
plt.boxplot(customer_df.TotalSales)
plt.subplot(1,3,2)
plt.boxplot(customer_df.OrderCount)
plt.subplot(1,3,3)
plt.boxplot(customer_df.AvgOrderValue)
plt.show()
```

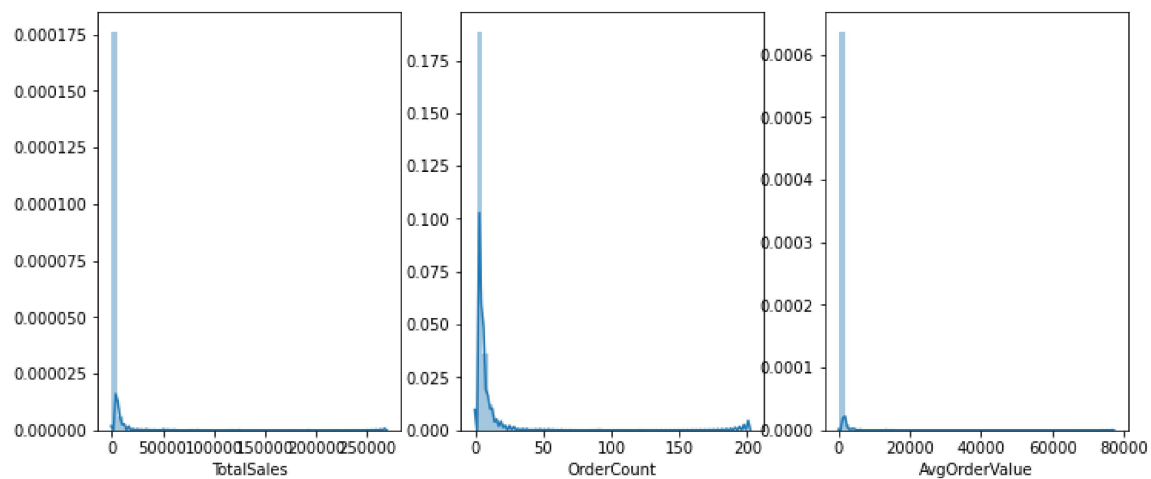


In [81]:

```
import seaborn as sns
```

In [82]:

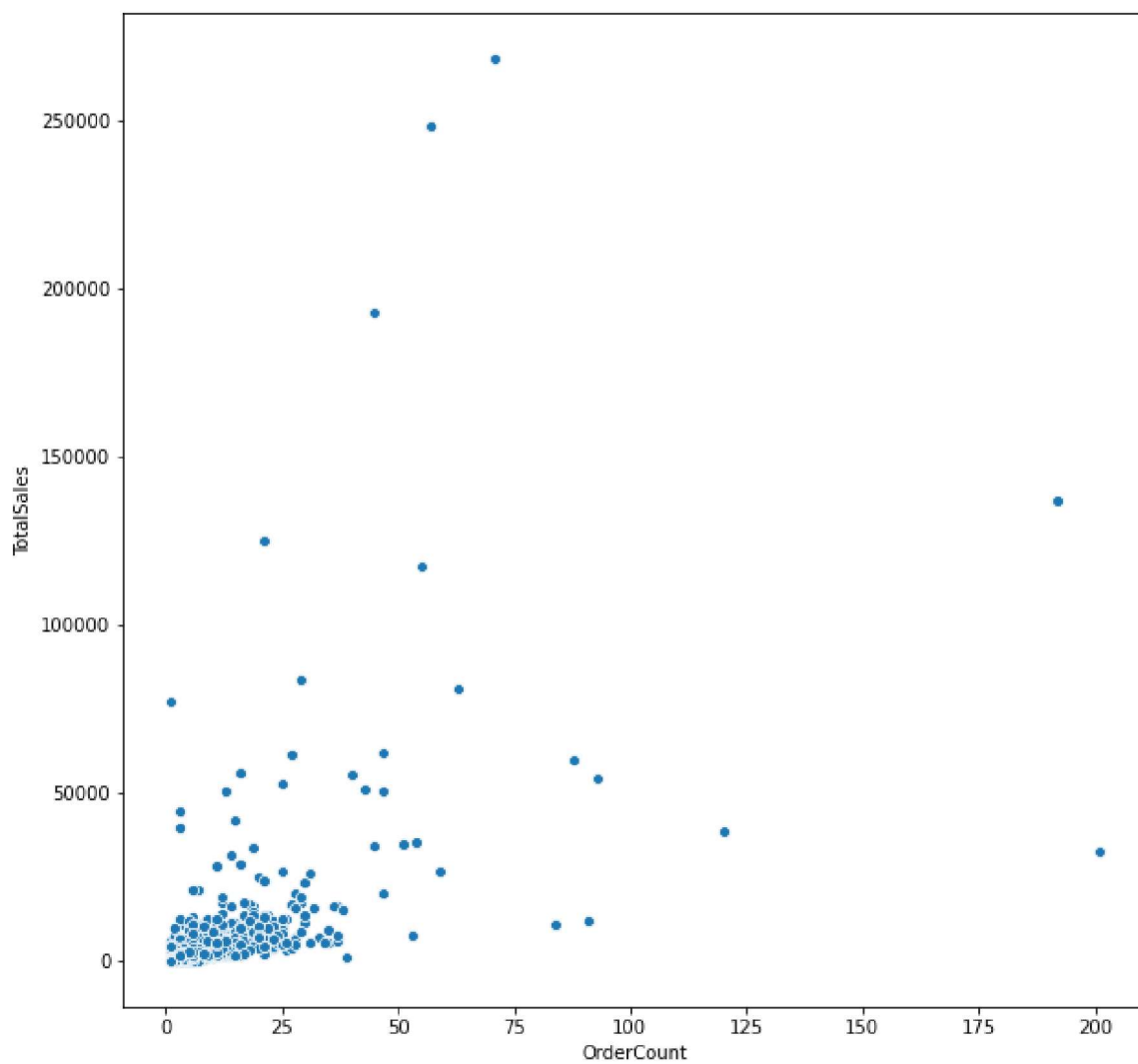
```
plt.figure(figsize=(12,5))
plt.subplot(1,3,1)
sns.distplot(customer_df.TotalSales)
plt.subplot(1,3,2)
sns.distplot(customer_df.OrderCount)
plt.subplot(1,3,3)
sns.distplot(customer_df.AvgOrderValue)
plt.show()
```





In [83]:

```
plt.figure(figsize=(10,10))  
sns.scatterplot(data=customer_df, y= 'TotalSales', x='OrderCount')  
plt.show()
```



In [84]:

```
# Theo như quan sát trên ta thấy các mẫu chủ yếu tập trung vào khoảng TotalSales ~[0, 1 5000], và OrderCount ~[1, 20]
customer_sub = customer_df[(customer_df.TotalSales<=15000) & (customer_df.OrderCount<=20)]
```

In [85]:

```
customer_sub.shape
```

Out[85]:

```
(4193, 3)
```

In [86]:

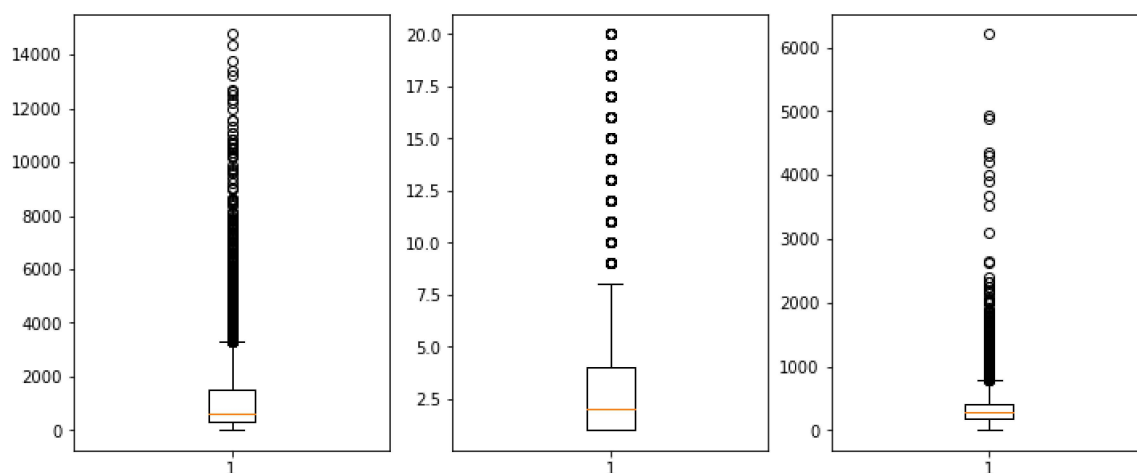
```
# Số mẫu đã xóa là các mẫu có TotalSales lớn và/hoặc OrderCount lớn
customer_df.shape[0]-customer_sub.shape[0]
```

Out[86]:

```
105
```

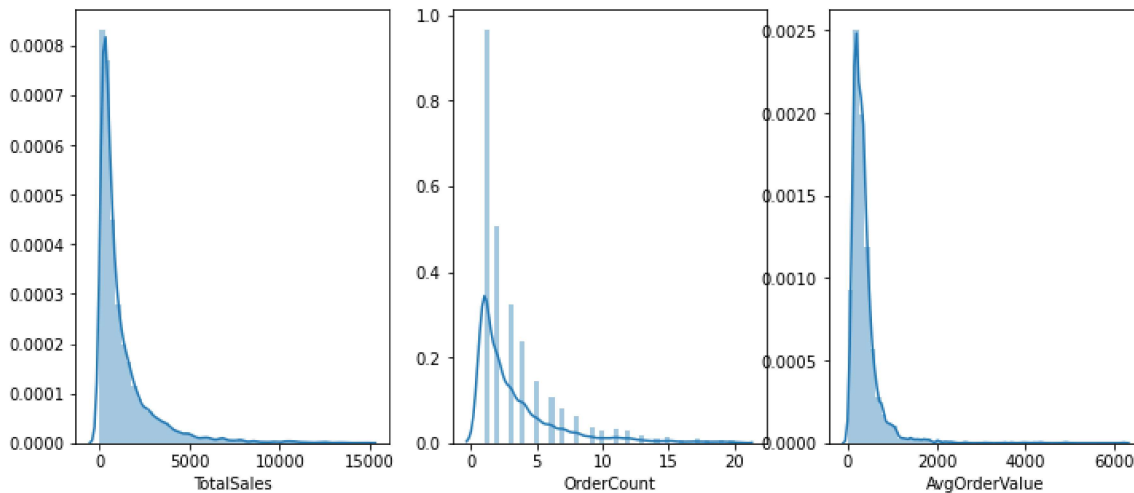
In [87]:

```
# Sau khi đã xóa một số mẫu, kết quả như sau:
plt.figure(figsize=(12,5))
plt.subplot(1,3,1)
plt.boxplot(customer_sub.TotalSales)
plt.subplot(1,3,2)
plt.boxplot(customer_sub.OrderCount)
plt.subplot(1,3,3)
plt.boxplot(customer_sub.AvgOrderValue)
plt.show()
```



In [88]:

```
plt.figure(figsize=(12,5))
plt.subplot(1,3,1)
sns.distplot(customer_sub.TotalSales)
plt.subplot(1,3,2)
sns.distplot(customer_sub.OrderCount)
plt.subplot(1,3,3)
sns.distplot(customer_sub.AvgOrderValue)
plt.show()
```



In [89]:

```
# Có outlier trên ở cả 3 features
# Áp dụng Scaler để chuẩn hóa
```

In [90]:

```
#rs = MinMaxScaler()
rs = StandardScaler()
rs.fit(customer_sub.drop('AvgOrderValue', axis=1))
data = rs.transform(customer_sub.drop('AvgOrderValue', axis=1))
```

In [91]:

```
X = pd.DataFrame(data, columns=['TotalSales', 'OrderCount'])
#X = X.drop('AvgOrderValue', axis=1)
```

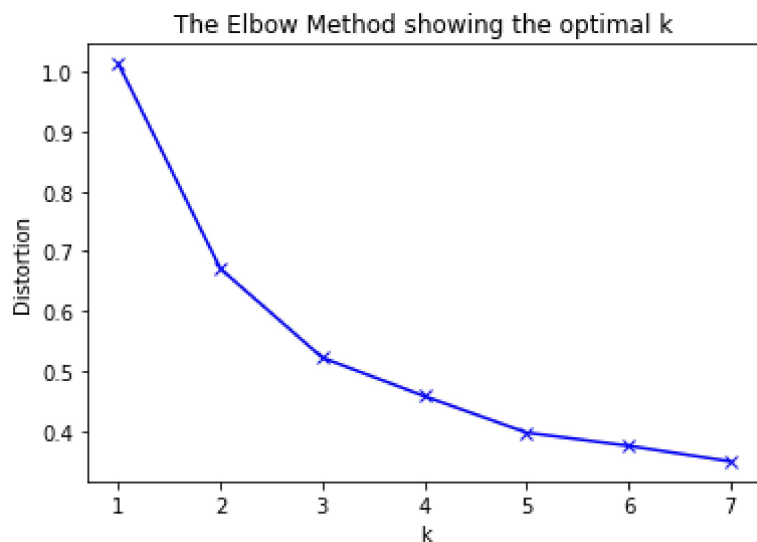
## 2. Tìm số cụm phù hợp k?

In [92]:

[illegible]

In [93]:

```
# Trực quan elbow
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



In [94]:

```
# Số cụm k=4
```

In [95]:

```
# Áp dụng k = 5
kmeans = KMeans(n_clusters=5)
kmeans.fit(X)
```

Out[95]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
        n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
        random_state=None, tol=0.0001, verbose=0)
```

In [96]:

```
centroids = kmeans.cluster_centers_  
labels = kmeans.labels_  
print("Centroids in min-max scale:")  
print(centroids)  
print("Centroids in normal:")  
normal_centroids = rs.inverse_transform(centroids)  
print(normal_centroids)  
print(labels)
```

Centroids in min-max scale:

```
[[ 0.94895332  0.86418038]  
 [-0.50655825 -0.60450607]  
 [ 4.68630016  2.8417935 ]  
 [-0.06424152  0.12253681]  
 [ 1.60515243  2.75193373]]
```

Centroids in normal:

```
[[2.82585532e+03  6.24000000e+00]  
 [3.96429380e+02  1.37155172e+00]  
 [9.06394205e+03  1.27954545e+01]  
 [1.13470982e+03  3.78157645e+00]  
 [3.92113150e+03  1.24975845e+01]]  
[0 3 1 ... 1 4 3]
```

In [97]:

```
X['Group'] = labels
```

In [98]:

```
customer_sub['Group'] = labels
```

c:\program files\python36\lib\site-packages\ipykernel\_launcher.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [http://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

"""Entry point for launching an IPython kernel.

In [99]:

```
customer_sub.head()
```

Out[99]:

CustomerID	TotalSales	OrderCount	AvgOrderValue	Group
12347.0	4085.18	6	680.863333	0
12348.0	1797.24	4	449.310000	3
12349.0	1757.55	1	1757.550000	1
12350.0	334.40	1	334.400000	1
12352.0	2506.04	8	313.255000	0

### 3. Vẽ hình, xem kết quả. Giải thích từng cụm.

In [100]:

```
from mpl_toolkits.mplot3d import Axes3D
```

In [101]:

```
# fig = plt.figure(figsize=(10,10))
# ax = fig.add_subplot(1, 1, 1, projection='3d')
# ax.scatter(customer_sub.TotalSales, customer_sub.AvgOrderValue, customer_sub.OrderCount,
#            c=customer_sub.Group, cmap=plt.cm.Spectral)
# ax.set_xlabel("TotalSales", fontsize=18)
# ax.set_ylabel("AvgOrderValue", fontsize=18)
# ax.set_zlabel("OrderCount", fontsize=18)
# ax.set_title("Customer Segmentation", fontsize=22)
```

In [102]:

```
X.head()
```

Out[102]:

	TotalSales	OrderCount	Group
0	1.703437	0.791779	0
1	0.332692	0.188430	3
2	0.308913	-0.716594	1
3	-0.543721	-0.716594	1
4	0.757346	1.395127	0

In [103]:

```
X.groupby('Group').count()
```

Out[103]:

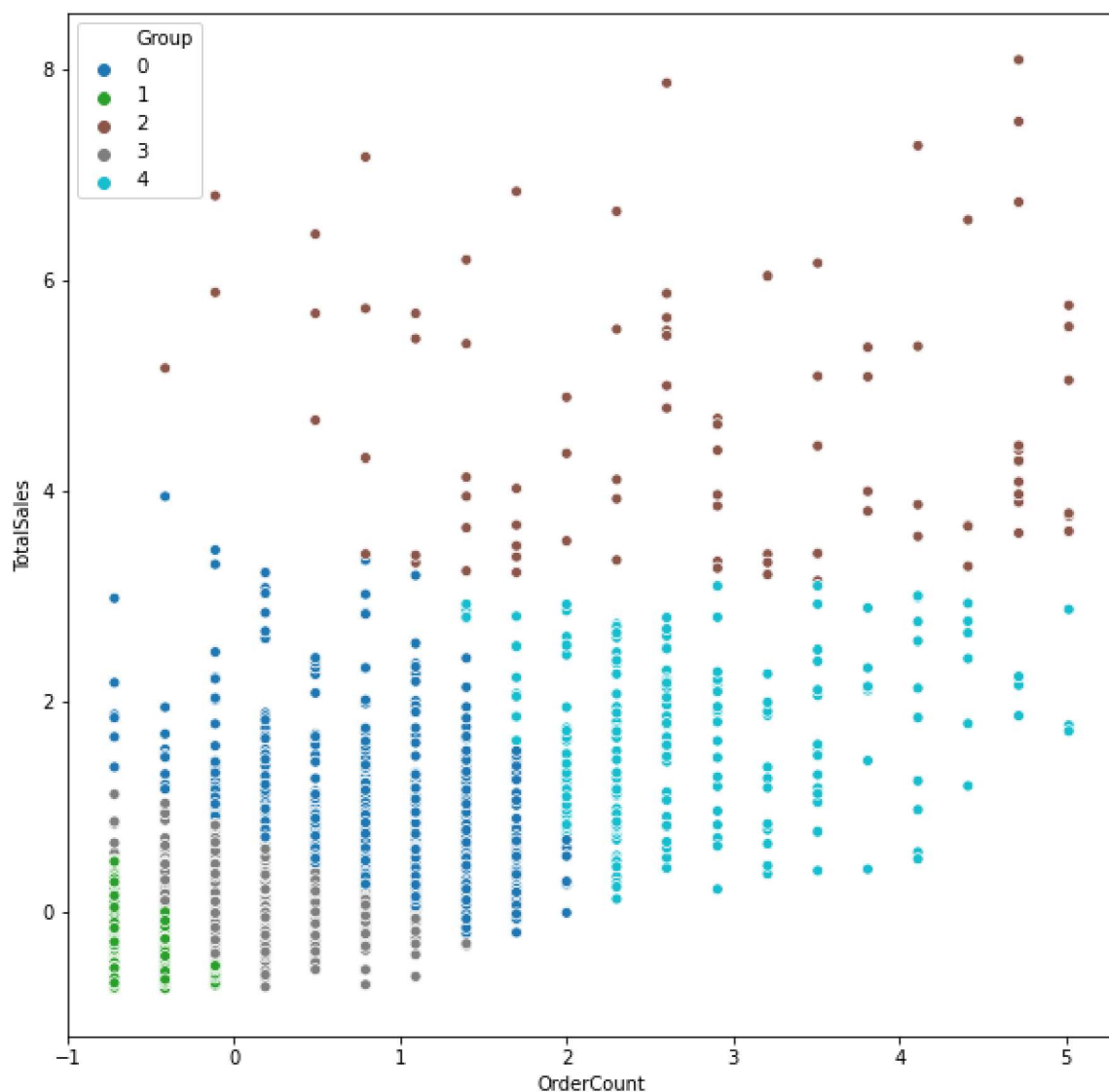
	TotalSales	OrderCount
Group		
0	525	525
1	2320	2320
2	88	88
3	1053	1053
4	207	207

In [104]:

```
import seaborn as sns
```

In [105]:

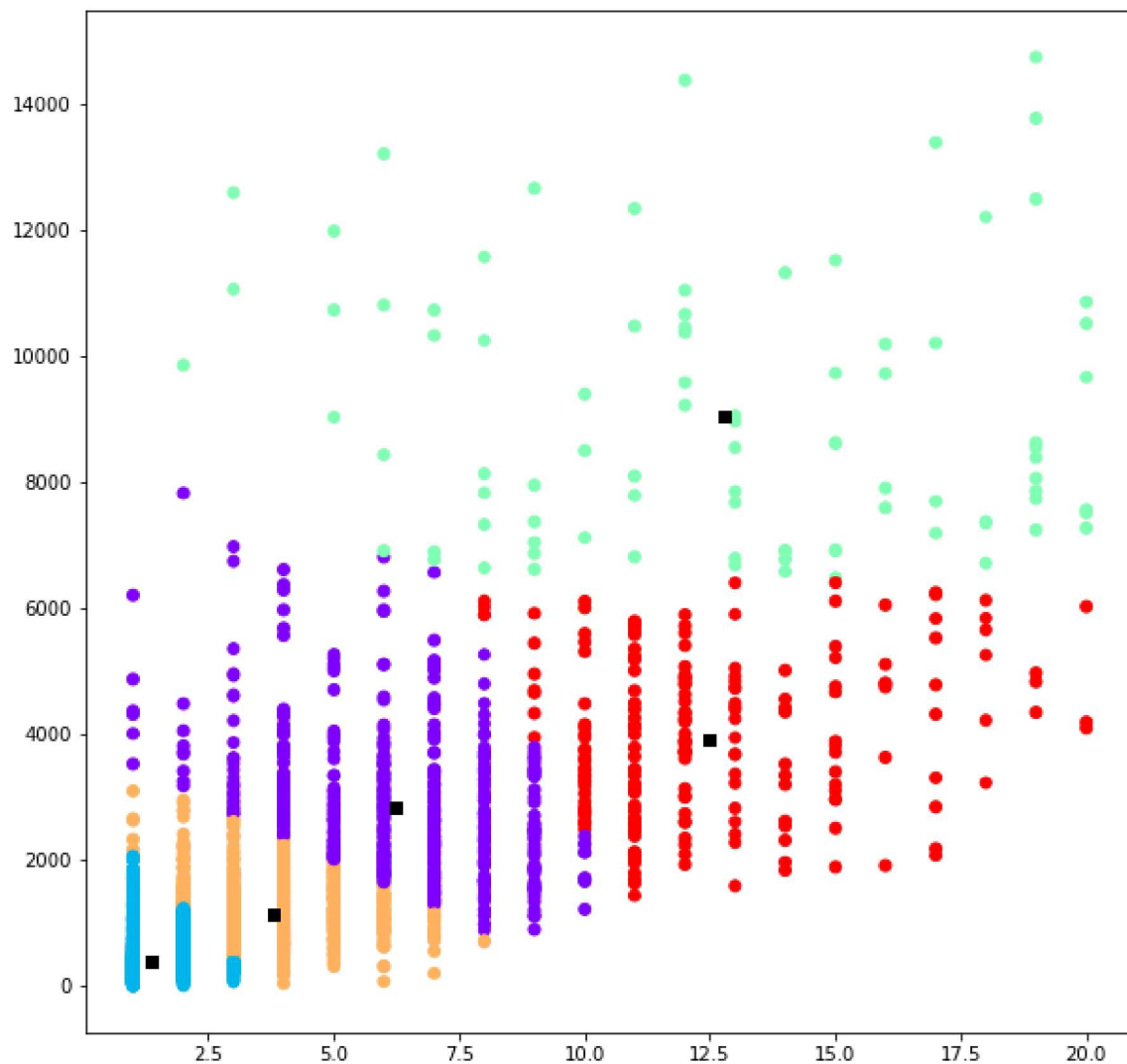
```
plt.figure(figsize=(10,10))
sns.scatterplot(data=X, x='OrderCount', y='TotalSales',
               hue='Group', palette='tab10',
               legend='full')
plt.show()
```





In [106]:

```
plt.figure(figsize=(10,10))
plt.scatter(customer_sub.OrderCount, customer_sub.TotalSales, c = customer_sub.Group, c
map='rainbow')
plt.scatter(normal_centroids[:, 1], normal_centroids[:, 0], marker = "s",c='black')
plt.show()
```



In [107]:

```
customer_sub.groupby('Group').count()
```

Out[107]:

	TotalSales	OrderCount	AvgOrderValue
Group			
0	525	525	525
1	2320	2320	2320
2	88	88	88
3	1053	1053	1053
4	207	207	207

In [108]:

```
i = 0
for cluster in normal_centroids:
    print("*** Centroid Cluster: " + str(i))
    print("TotalSales:",round(cluster[0],2))
    print("OrderCount:",round(cluster[1],2))
    #print("AvgTotalSales:",round(cluster[2],2))
    i = i+1
```

```
*** Centroid Cluster: 0
TotalSales: 2825.86
OrderCount: 6.24
*** Centroid Cluster: 1
TotalSales: 396.43
OrderCount: 1.37
*** Centroid Cluster: 2
TotalSales: 9063.94
OrderCount: 12.8
*** Centroid Cluster: 3
TotalSales: 1134.71
OrderCount: 3.78
*** Centroid Cluster: 4
TotalSales: 3921.13
OrderCount: 12.5
```

## Nhận xét:

- Cụm [396, 1.4] có giá trị thấp nhất cho các thuộc tính, nhưng lại có số lượng khách hàng nhiều nhất. Từ đó cho thấy cụm này chứa rất nhiều khách hàng có TotalSales và OrderCount thấp. Xét về từng khách hàng, họ đem lại giá trị thấp nhưng xét về số lượng khách hàng thì lại rất lớn.
- Mặt khác, cụm [9063, 12.8], có giá trị rất cao cho các thuộc tính đặc biệt là TotalSales, nhưng lại có số lượng khách hàng ít nhất. Xét về từng khách hàng trong nhóm này, họ đem lại giá trị cao cho doanh nghiệp, vì họ mua những mặt hàng có giá trị cao và đem lại doanh thu cao nhất. Có thể tập trung nỗ lực tiếp thị vào phân khúc khách hàng này, vì nó sẽ mang lại lợi nhuận cao nhất.
- Những khách hàng ở cụm [3921, 12.5] khá thú vị. Họ mua hàng thường xuyên nhất, TotalSales cũng cao (chỉ kém nhóm vừa nói ở trên. Tuy nhiên, nhóm này khá ít khách hàng. Vì vậy, nên tiếp thị các mặt hàng với giá mỗi mặt hàng vừa phải cho phân khúc khách hàng này.
- Những khách hàng ở cụm [1134, 3.8] có đóng góp của họ vào doanh thu và số lượng đơn đặt hàng ở mức khá thấp, tuy nhiên số lượng khách hàng nhiều (chỉ ít hơn một nhóm). Đây là những khách hàng không thường xuyên mua hàng, cũng không thường xuyên mua đắt tiền.
- Chúng ta cũng có thể tìm hiểu những mặt hàng bán chạy nhất cho từng phân khúc khách hàng.

In [113]:

```
high_value_cluster = customer_sub.loc[customer_sub['Group'] == 2]

pd.DataFrame(
    df.loc[
        df['CustomerID'].isin(high_value_cluster.index)
    ].groupby(['Description', 'UnitPrice']).count()[
        'StockCode'
    ].sort_values(ascending=False).head()
)
```

Out[113]:

		StockCode
Description	UnitPrice	
REGENCY CAKESTAND 3 TIER	12.75	123
ALARM CLOCK BAKELIKE RED	3.75	114
LUNCH BAG RED RETROSPOT	1.65	108
LUNCH BAG PINK POLKADOT	1.65	103
ALARM CLOCK BAKELIKE GREEN	3.75	98

In [112]:

```
high_value_cluster = customer_sub.loc[customer_sub['Group'] == 1]

pd.DataFrame(
    df.loc[
        df['CustomerID'].isin(high_value_cluster.index)
    ].groupby(['Description', 'UnitPrice']).count()[
        'StockCode'
    ].sort_values(ascending=False).head()
)
```

Out[112]:

		StockCode
Description	UnitPrice	
WHITE HANGING HEART T-LIGHT HOLDER	2.95	301
REGENCY CAKESTAND 3 TIER	12.75	263
REX CASH+CARRY JUMBO SHOPPER	0.95	246
ASSORTED COLOUR BIRD ORNAMENT	1.69	242
BAKING SET 9 PIECE RETROSPOT	4.95	218

In [109]:

```
high_value_cluster = customer_sub.loc[customer_sub['Group'] == 3]

pd.DataFrame(
    df.loc[
        df['CustomerID'].isin(high_value_cluster.index)
    ].groupby(['Description', 'UnitPrice']).count()[
        'StockCode'
    ].sort_values(ascending=False).head()
)
```

Out[109]:

		StockCode
Description	UnitPrice	
WHITE HANGING HEART T-LIGHT HOLDER	2.95	417
ASSORTED COLOUR BIRD ORNAMENT	1.69	359
REGENCY CAKESTAND 3 TIER	12.75	323
REX CASH+CARRY JUMBO SHOPPER	0.95	261
NATURAL SLATE HEART CHALKBOARD	2.95	250

In [110]:

```
high_value_cluster = customer_sub.loc[customer_sub['Group'] == 0]

pd.DataFrame(
    df.loc[
        df['CustomerID'].isin(high_value_cluster.index)
    ].groupby(['Description', 'UnitPrice']).count()[
        'StockCode'
    ].sort_values(ascending=False).head()
)
```

Out[110]:

		StockCode
	Description	UnitPrice
	WHITE HANGING HEART T-LIGHT HOLDER	2.95
	REGENCY CAKESTAND 3 TIER	12.75
	LUNCH BAG RED RETROSPOT	1.65
	ASSORTED COLOUR BIRD ORNAMENT	1.69
	SET OF 3 CAKE TINS PANTRY DESIGN	4.95

In [111]:

```
high_value_cluster = customer_sub.loc[customer_sub['Group'] == 4]

pd.DataFrame(
    df.loc[
        df['CustomerID'].isin(high_value_cluster.index)
    ].groupby(['Description', 'UnitPrice']).count()[
        'StockCode'
    ].sort_values(ascending=False).head()
)
```

Out[111]:

		StockCode
	Description	UnitPrice
	WHITE HANGING HEART T-LIGHT HOLDER	2.95
	JUMBO BAG RED RETROSPOT	2.08
	LUNCH BAG RED RETROSPOT	1.65
	PARTY BUNTING	4.95
	LUNCH BAG BLACK SKULL.	1.65

In [114]:

```
# 10 sản phẩm bán chạy nhất
pd.DataFrame(
    df.loc[
        df['CustomerID'].isin(customer_sub.index)
    ].groupby(['Description', 'UnitPrice']).count()[
        'StockCode'
    ].sort_values(ascending=False).head(10)
)
```

Out[114]:

		StockCode
Description	UnitPrice	
WHITE HANGING HEART T-LIGHT HOLDER	2.95	1446
REGENCY CAKESTAND 3 TIER	12.75	1194
ASSORTED COLOUR BIRD ORNAMENT	1.69	1163
PARTY BUNTING	4.95	1011
LUNCH BAG RED RETROSPOT	1.65	959
SET OF 3 CAKE TINS PANTRY DESIGN	4.95	889
JUMBO BAG RED RETROSPOT	2.08	875
LUNCH BAG BLACK SKULL.	1.65	840
REX CASH+CARRY JUMBO SHOPPER	0.95	832
PACK OF 72 RETROSPOT CAKE CASES	0.55	817