Chapter 15 - Ex3: 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 GMM 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
```

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 [3]:
```
df.shape
```

Out[3]:
```
(541909, 8)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
InvoiceNo      541909 non-null object
StockCode      541909 non-null object
Description    540455 non-null object
Quantity       541909 non-null int64
InvoiceDate    541909 non-null datetime64[ns]
UnitPrice      541909 non-null float64
CustomerID     406829 non-null float64
Country        541909 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

In [5]:
```
df.head()
```

Out[5]:

	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	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [6]:
```
# Loại bỏ các dòng có Quantity ấm (là hàng bị hủy)
df.loc[df['Quantity'] <= 0].shape
```

Out[6]:
```
(10624, 8)
```

In [7]:
```
df = df.loc[df['Quantity'] > 0]
```

In [8]:
```
df.shape
```

Out[8]:
```
(531285, 8)
```

In [9]:
```
# Loại bỏ các dòng có CustomerID = NULL
df = df[pd.notnull(df['CustomerID'])]
df.shape
```

Out[9]:
```
(397924, 8)
```

In [10]:
```
df.head()
```

Out[10]:

	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	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [11]:
```
# 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
dtype: int64
```

In [12]:
```
# 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 ~ 2011-12-09 12:50:00
```

In [13]:
```
df.loc[df['InvoiceDate'] >= '2011-12-01'].shape
```

Out[13]: (17304, 8)

In [14]:
```
df = df.loc[df['InvoiceDate'] < '2011-12-01']
```

In [15]:
```
df.shape
```

Out[15]: (380620, 8)

In [16]:
```
# Tính total sales (Sales = Quantity * UnitPrice)
df['Sales'] = df['Quantity'] * df['UnitPrice']
df.head()
```

Out[16]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sale
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.3
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.3
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.3
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.3

In [17]:
```
# 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 [18]:
```
customer_df.head()
```

Out[18]:

CustomerID	Sales	InvoiceNo
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 [19]:
```
# 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[19]:

CustomerID	TotalSales	OrderCount	AvgOrderValue
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 [20]:
```
# Thống kê chung
customer_df.describe()
```

Out[20]:

	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 [21]:
```
import numpy as np
np.ptp(customer_df.TotalSales)
```

```
c:\program files\python36\lib\site-packages\numpy\core\fromnumeric.py:2542: Fut
ureWarning: Method .ptp is deprecated and will be removed in a future version.
Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[21]: 268477.9999999998

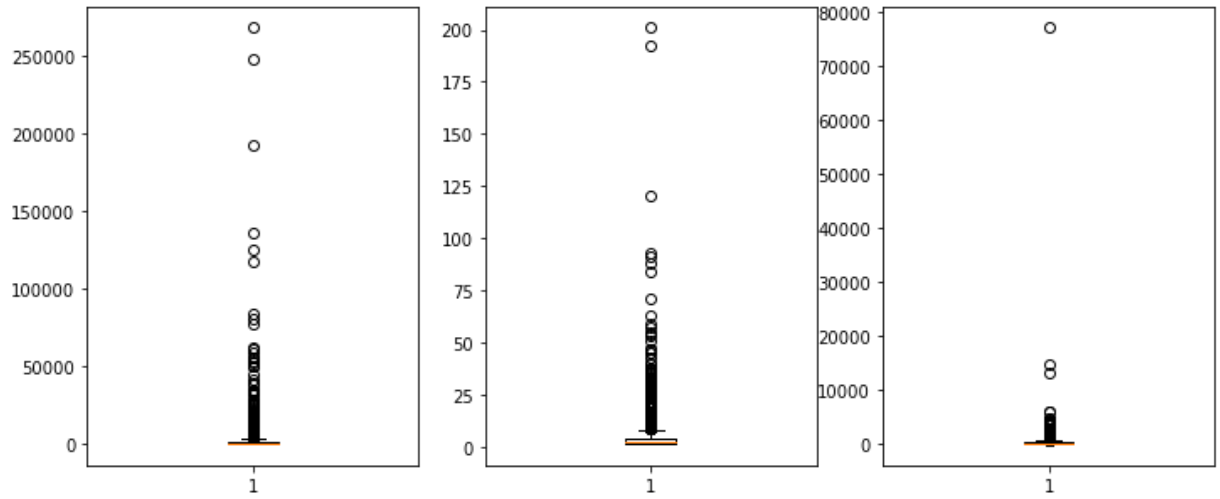
In [22]:
```
np.ptp(customer_df.OrderCount)
```

Out[22]: 200

In [23]:
```
np.ptp(customer_df.AvgOrderValue)
```

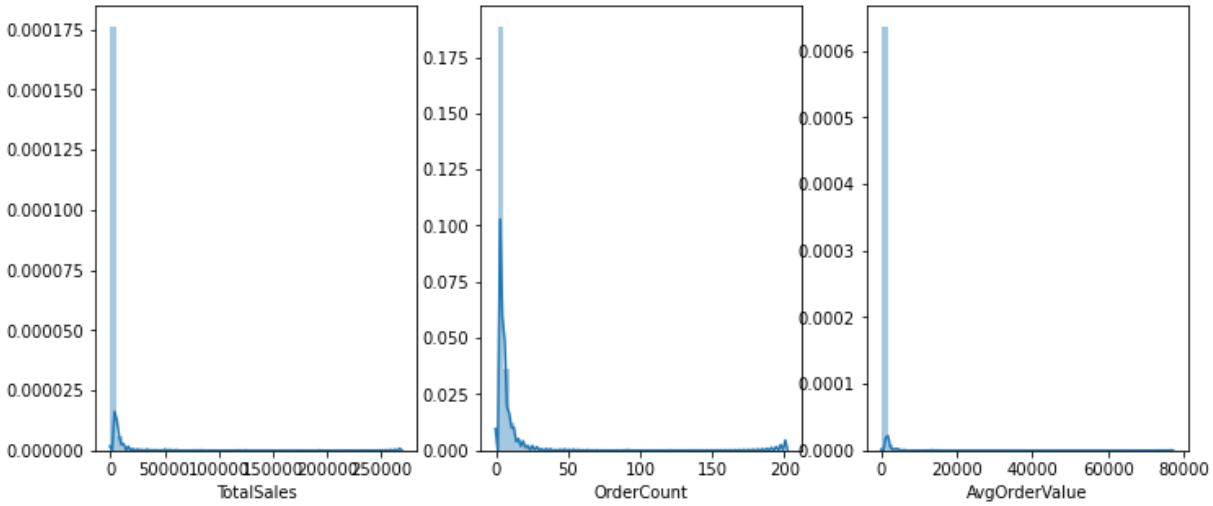
Out[23]: 77183.6

In [24]:
```
# => 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()
```

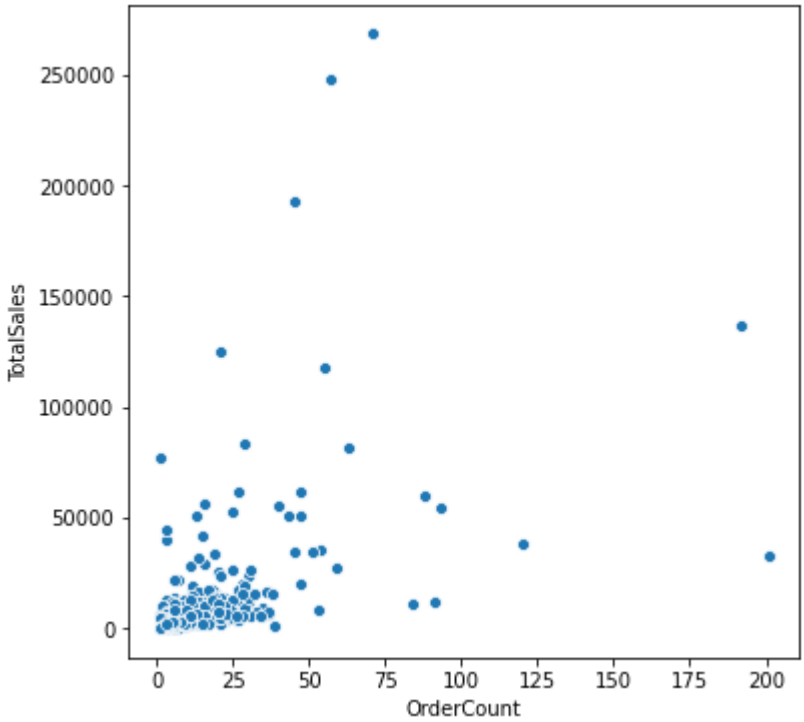


In [25]:
```
import seaborn as sns
```

In [26]:
```
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 [27]:
```
plt.figure(figsize=(6,6))
sns.scatterplot(data=customer_df,
                y= 'TotalSales', x='OrderCount')
plt.show()
```



In [28]:
```
# Theo như quan sát trên ta thấy các mẫu chủ yếu tập trung vào khoảng
# TotalSales ~[0, 15000], và OrderCount ~[1, 20]
customer_sub = customer_df[(customer_df.TotalSales<=15000) & (customer_df.OrderC
```

In [29]:
```
customer_sub.shape
```

Out[29]:
```
(4193, 3)
```

In [30]:
```
# số mẫu đã xóa
customer_df.shape[0]-customer_sub.shape[0]
```

Out[30]:
```
105
```

In [31]:
```
# Có outlier trên ở cả 3 features
# Áp dụng RobustScaler để chuẩn hóa
from sklearn.preprocessing import RobustScaler, MinMaxScaler, StandardScaler
```

In [32]:
```
# #rs = RobustScaler()
# rs = MinMaxScaler()
# rs = StandardScaler()
# rs.fit(customer_df)
# data = rs.transform(customer_df)
```

In [33]:
```
# data[:5]
```

In [34]:
```
# X = pd.DataFrame(data, columns=['TotalSales', 'OrderCount', 'AvgOrderValue'])
# X.head()
```

In [35]:
```
#X = X.drop('AvgOrderValue', axis=1)
```

In [36]:
```
#X.head()
```

In [37]:
```
#X = X.dropna()
```

- From this data, the three columns, TotalSales, OrderCount, and AvgOrderValue, have different scales. TotalSales can take any values from 0 to 268478, while OrderCount takes values between 1 and 201. Clustering algorithms are highly affected by the scales of the data, so we need to normalize this data to be on the same scale. We are going to take two steps to normalize this data. First, we are going to rank the data, so that the values of each column range from 1 to 4298, which is the total number of records. Take a look at the following code:

In [38]:
```
# rank_df = customer_df.rank(method='first')
```

In [39]:
```
# X = ((rank_df - rank_df.mean()) / rank_df.std())
```

In [40]:
```
#rs = MinMaxScaler()
rs = StandardScaler()
rs.fit(customer_sub)
data = rs.transform(customer_sub)
```

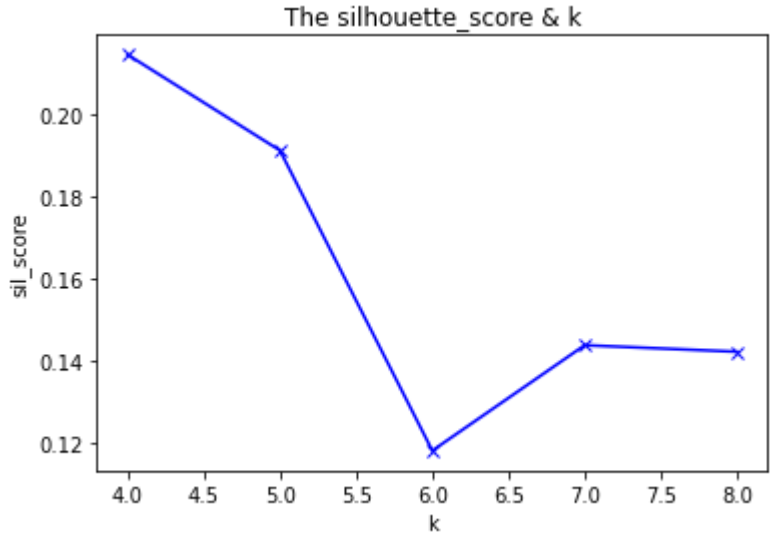
In [41]:
```
X = pd.DataFrame(data, columns=['TotalSales', 'OrderCount', 'AvgOrderValue'])
#X = X.drop('AvgOrderValue', axis=1)
```

2. Tìm số cụm phù hợp k?

In [42]:
```
from sklearn.mixture import GaussianMixture
from sklearn import metrics
```

In [43]:
```
list_sil = [] # Chứa danh sách các giá trị sil
K = range(4,9) # Chứa danh sách số cụm có thể
for k in K:
    gmm = GaussianMixture(n_components=k) # 4,5,6,7,8...
    gmm.fit(X)
    labels = gmm.predict(X)
    # k = 2 => 0 , 1
    # k = 3 => 0, 1, 2
    sil = metrics.silhouette_score(X, labels, metric='euclidean')
    list_sil.append(sil)
```

In [44]:
```
# Plot
plt.plot(K, list_sil, 'bx-')
plt.xlabel('k')
plt.ylabel('sil_score')
plt.title('The silhouette_score & k')
plt.show()
```



In [45]:
```
# Số cụm k=4 được đề xuất vì có sil lớn nhất
```

In [46]:
```
gmm = GaussianMixture(n_components=4)
gmm.fit(X)
```

Out[46]:
```
GaussianMixture(covariance_type='full', init_params='kmeans', max_iter=100,
                means_init=None, n_components=4, n_init=1, precisions_init=Non
e,
                random_state=None, reg_covar=1e-06, tol=0.001, verbose=0,
                verbose_interval=10, warm_start=False, weights_init=None)
```

In [47]:
```
# Sau khi model đã hội tụ, weights, means, và covariances cần phải được giải quyế
# In các thông số này:
```

In [48]:
```
print(gmm.weights_)
```

```
[0.36258293 0.36483781 0.21621886 0.0563604 ]
```

In [49]:
```
print(gmm.means_)
```

```
[[-0.54660253 -0.71659373 -0.09306337]
 [-0.30220964 -0.15898894 -0.28917158]
 [ 1.09648045  1.49735661  0.02831373]
 [ 1.26625234 -0.10516101  2.36197663]]
```

In [50]:
```
print(gmm.covariances_)
```

```
[[[ 2.70354797e-02  1.77652067e-30  1.29505142e-01]
  [ 1.77859620e-30  1.00000000e-06  3.01470174e-31]
  [ 1.29505142e-01  3.03545701e-31  6.20378458e-01]]

 [[ 5.68365916e-02  3.70095958e-02  6.62801489e-02]
  [ 3.70095958e-02  8.26230267e-02 -5.75950504e-03]
  [ 6.62801489e-02 -5.75950504e-03  1.26910663e-01]]

 [[ 1.66411701e+00  9.58030924e-01  4.37912809e-01]
  [ 9.58030924e-01  1.28754059e+00 -2.50940505e-02]
  [ 4.37912809e-01 -2.50940505e-02  2.55624702e-01]]

 [[ 2.08792192e+00  4.19466844e-01  1.75056783e+00]
  [ 4.19466844e-01  1.89008513e-01 -2.00268287e-01]
  [ 1.75056783e+00 -2.00268287e-01  5.77067073e+00]]]
```

In [51]:
```
types = gmm.predict(X)
```

In [52]:
```
types
```

Out[52]:
```
array([2, 2, 0, ..., 0, 2, 1], dtype=int64)
```

In [53]:
```
X['Group'] = types
```

In [54]:
```
customer_sub['Group'] = types
```

```
c:\program files\python36\lib\site-packages\ipykernel_launcher.py:1: SettingWit
hCopyWarning:
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/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydat
a.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y)
  """Entry point for launching an IPython kernel.
```

In [55]:
```
customer_df.head()
```

Out[55]:

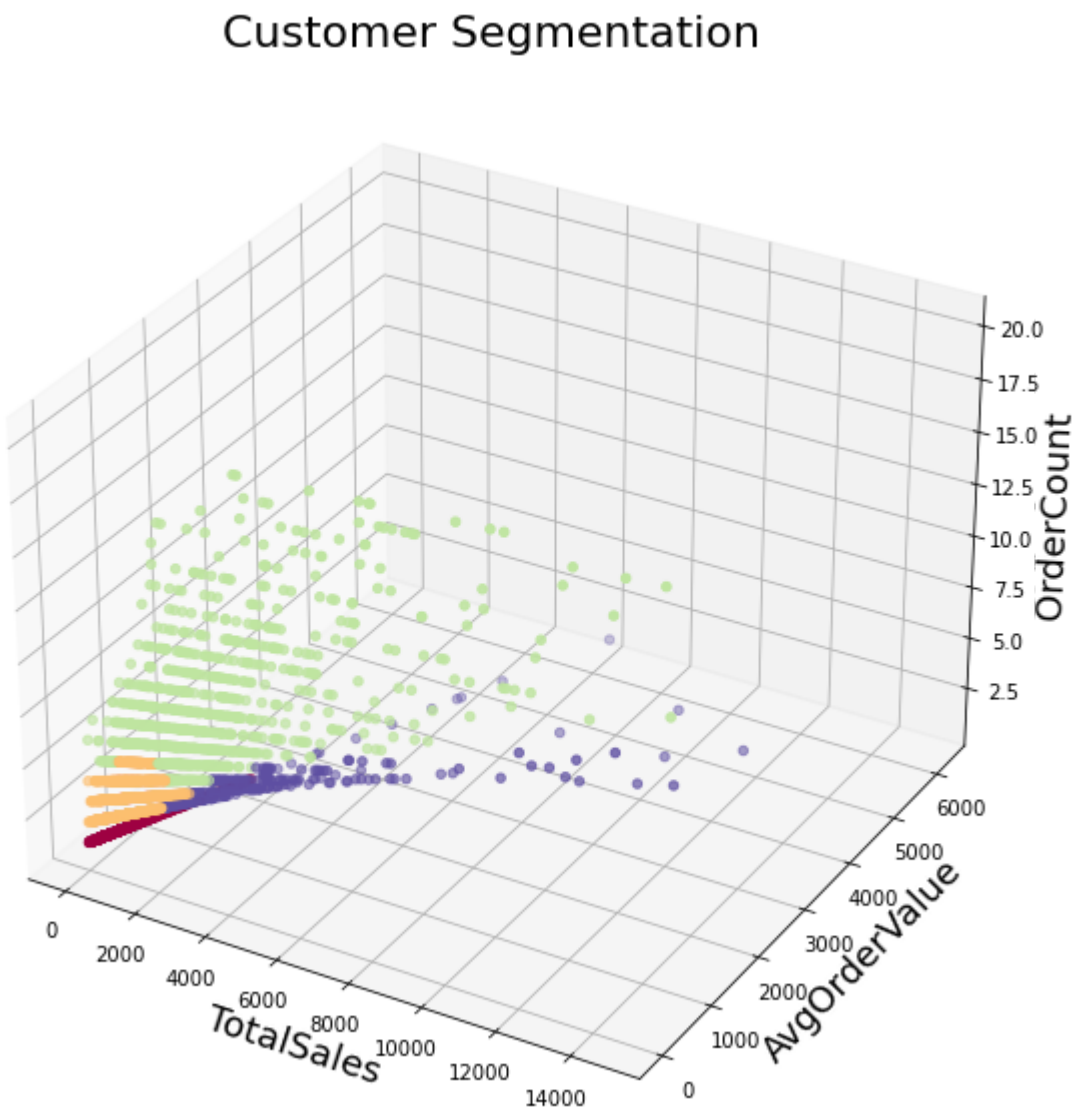
CustomerID	TotalSales	OrderCount	AvgOrderValue
12346.0	77183.60	1	77183.600000
12347.0	4085.18	6	680.863333
12348.0	1797.24	4	449.310000
12349.0	1757.55	1	1757.550000
12350.0	334.40	1	334.400000

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

In [56]:
```
from mpl_toolkits.mplot3d import Axes3D
```

In [57]:
```
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)
```

Out[57]:
```
Text(0.5, 0.92, 'Customer Segmentation')
```



In [58]:
```
X.head()
```

Out[58]:

	TotalSales	OrderCount	AvgOrderValue	Group
0	1.703437	0.791779	0.915087	2
1	0.332692	0.188430	0.250531	2
2	0.308913	-0.716594	4.005172	0
3	-0.543721	-0.716594	-0.079260	0
4	0.757346	1.395127	-0.139946	2

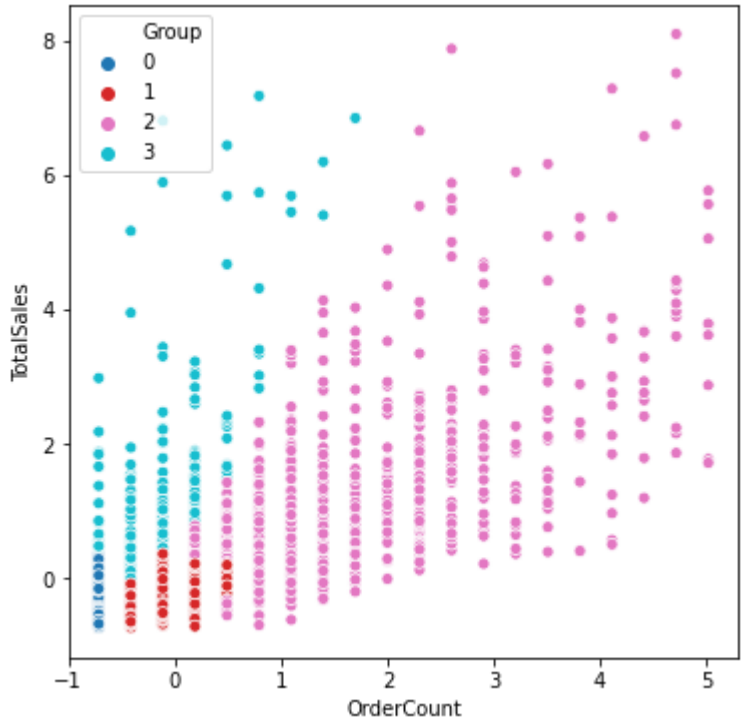
In [59]:
```
X.groupby('Group').count()
```

Out[59]:

Group	TotalSales	OrderCount	AvgOrderValue
0	1521	1521	1521
1	1541	1541	1541
2	900	900	900
3	231	231	231

In [60]:
```
import seaborn as sns
```

In [61]:
```
plt.figure(figsize=(6,6))
sns.scatterplot(data=X, x='OrderCount', y='TotalSales',
                hue='Group', palette='tab10',
                legend='full', markers='Group')
plt.show()
```



In [62]:
```
plt.figure(figsize=(6,6))
sns.scatterplot(data=customer_sub, x='OrderCount',
                y='TotalSales',
                hue='Group', palette='tab10',
                legend='full', markers='Group')
plt.show()
```