Chapter 11 - Ex2: Customer Segmentation

Step 1: Business Understanding

- Bạn vừa được tuyển dụng vào vị trí trưởng phòng Marketing của công ty X là công ty chuyên về hóa mỹ phẩm. Ngay tuần đầu tiên đi làm, giám đốc mời bạn lên và nói rằng: “Chiến lược quảng bá các sản phẩm của công ty hiện tại chưa đem lại hiệu quả cao. Nhiệm vụ của anh/chị là đưa ra giải pháp giúp cải thiện hiệu quả quảng bá, từ đó giúp tăng doanh thu bán hàng, cải thiện mức độ hài lòng của khách hàng”
- Sau khi đặt ra các câu hỏi cụ thể cho sếp và tự bản thân tìm hiểu => bạn xác định được vấn đề: Hiện tại khách hàng của công ty chưa được phân khúc cụ thể => cần phân khúc khách hàng => Cải thiện hiệu quả quảng bá => tăng doanh thu bán hàng, cải thiện mức độ hài lòng của khách hàng

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
#%cd '/content/gdrive/My Drive/Workshop_CustumerSegmentation/source'
```

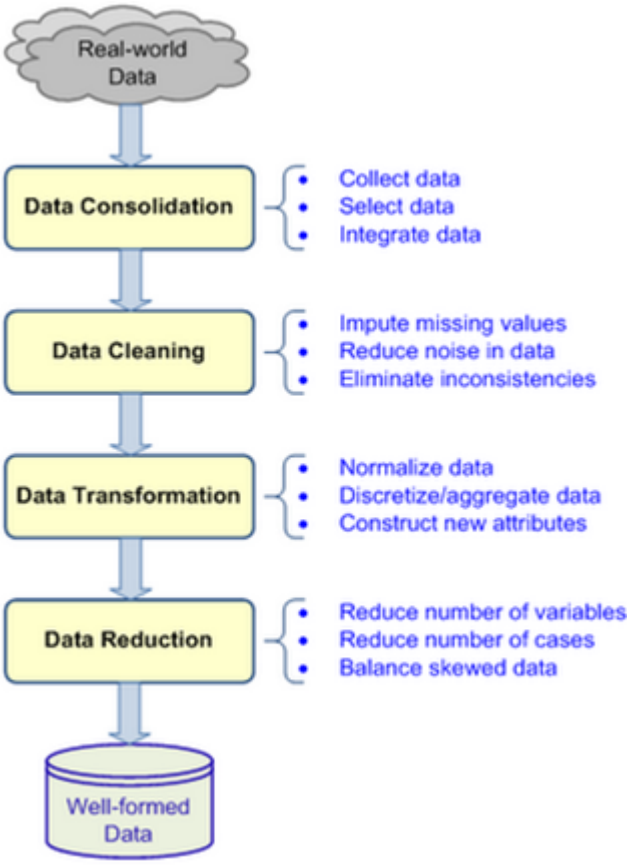
In [3]:
```
# Tải các thư viện cần thiết
import pandas as pd
import numpy as np
import numpy as numpy
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

In [4]:
```
# Hiển thị số thập phân thay cho science math number
numpy.set_printoptions(suppress=True)
```

Step 2: Data Understading/ Acquire

- Từ yêu cầu đã xác định: bạn xem xét các dữ liệu mà công ty đang có và thấy rằng công ty hiện tại có lưu trữ thông tin khách hàng trên hệ thống máy chủ. Và bạn liên hệ bộ phận quản lý dữ liệu để lấy về.
- Dữ liệu khách hàng hiện có 1000 mẫu chứa thông tin income, age, gender

Step 3: Data preparation/ Prepare



- Data Consolidation: hợp nhất dữ liệu từ các nguồn, lựa chọn các dữ liệu phù hợp

In [5]:
```
data = pd.read_csv('ageinc_g.csv', index_col=0)
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 3 columns):
income    1000 non-null int64
age       1000 non-null int64
gender    1000 non-null object
dtypes: int64(2), object(1)
memory usage: 31.2+ KB
```

In [7]:
```
data.head()
```

Out[7]:

	income	age	gender
0	101743	58	Female
1	49597	27	Female
2	36517	52	Male
3	33223	49	Male
4	72994	53	Female

- Data Cleaning: kiểm tra xem có dữ liệu không liên quan, dữ liệu null, dữ liệu outlier, hay

không? => Nếu có thì xử lý.

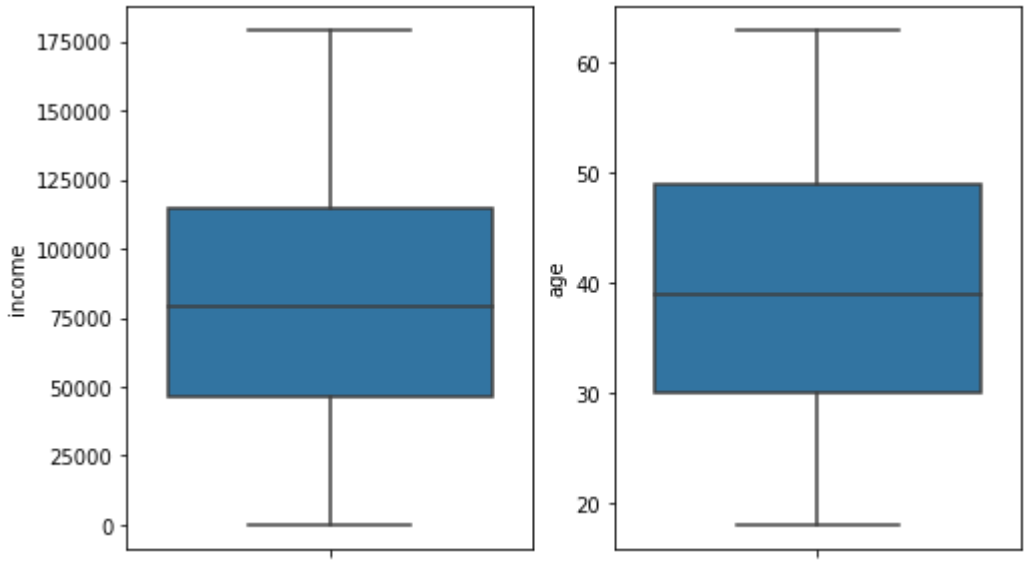
* Dataset có 3 cột là income, age, gender đều cần thiết để giải quyết bà
i toán => dữ liệu đã liên quan.
* Kiểm tra dữ liệu null:

In [8]:
```
# kiểm tra dữ liệu null
print(data.isnull().sum())
# => Không có dữ liệu null
```

```
income    0
age       0
gender    0
dtype: int64
```

* Phân tích đơn biến: trực quan hóa, kiểm tra dữ liệu outlier

In [9]:
```
# Trực quan hóa dữ liệu cho từng biến liên tục
plt.figure(figsize=(8,5))
plt.subplot(1,2,1)
sns.boxplot(data.income, orient="v")
plt.subplot(1,2,2)
sns.boxplot(data.age, orient="v")
plt.show()
# => Cả hai biến liên tục income và age đều không có outlier
```



- Data Transformation: chuẩn hóa dữ liệu, bổ sung thuộc tính mới

* Dataset có cột gender: là thuộc tính phân loại kiểu chuỗi => tạo cột p
hân loại kiểu số

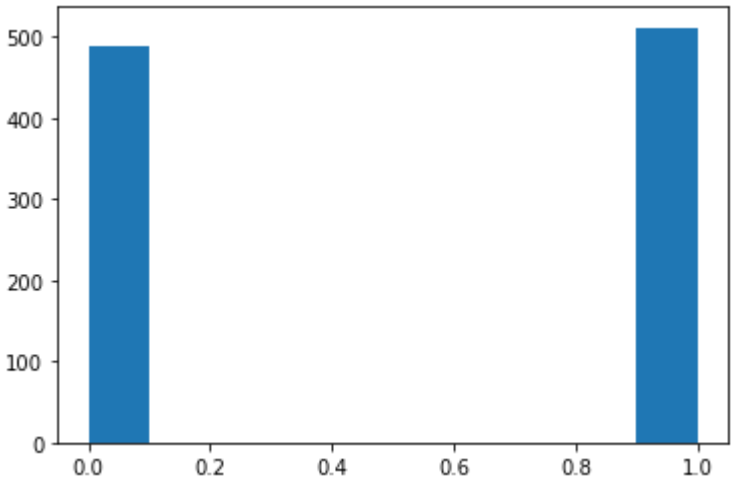
In [10]:
```
data = pd.get_dummies(data, drop_first=True)
```

In [11]:
```
data.head()
```

Out[11]:

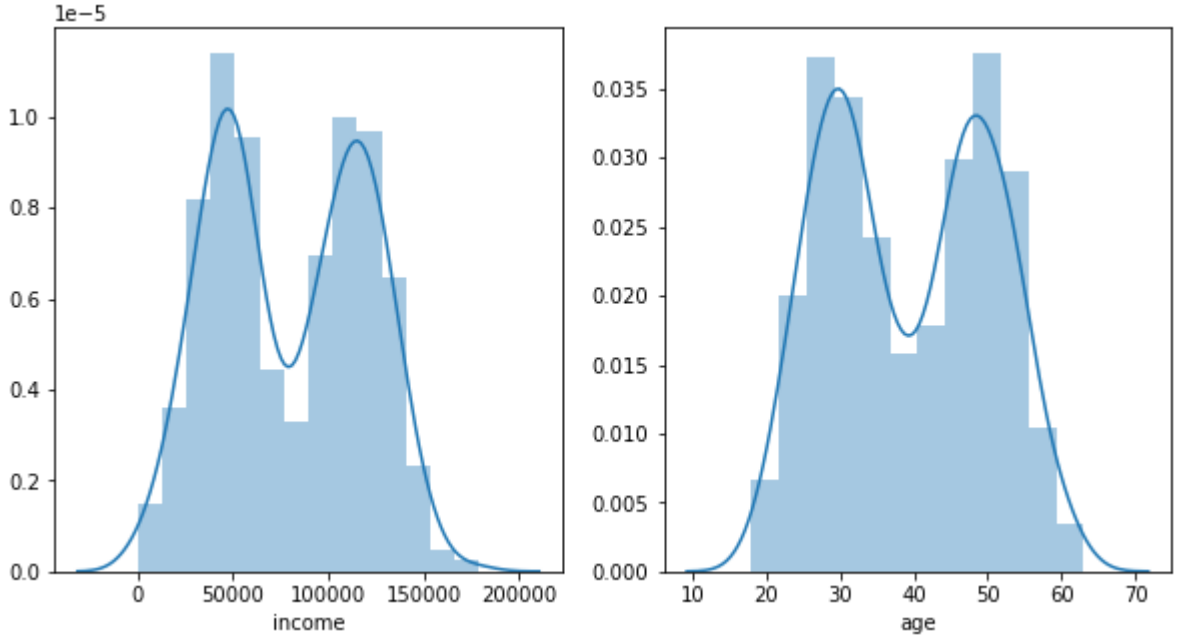
	income	age	gender_Male
0	101743	58	0
1	49597	27	0
2	36517	52	1
3	33223	49	1
4	72994	53	0

In [12]:
```
plt.hist(data.gender_Male)
plt.show()
```



* Customer Segmentation là một bài toán phân nhóm dựa trên sự tương tự về các thuộc tính và sẽ tính khoảng cách để biết mẫu này "gần" mẫu kia hay không bằng công thức tính khoảng cách Euclide
* Đối với các thuật toán cần tính khoảng cách thì dữ liệu trên các cột phải đảm bảo tính công bằng khi tính toán. Tuy nhiên, quan sát thấy income và age có range dữ liệu khác biệt, chênh lệch nhiều => cần phải chuẩn hóa dữ liệu.
* Muốn chuẩn hóa dữ liệu: cần kiểm tra phân phối của dữ liệu.

In [13]:
```
plt.figure(figsize=(10,5))
plt.subplot(1, 2, 1)
sns.distplot(data.income)
plt.subplot(1, 2, 2)
sns.distplot(data.age)
plt.show()
```



In [14]:
```
data.skew()
```

Out[14]:
```
income         0.028753
age            0.049110
gender_Male   -0.044077
dtype: float64
```

In [15]:
```
data.kurtosis()
```

Out[15]:
```
income        -1.210079
age           -1.262649
gender_Male   -2.002065
dtype: float64
```

- Từ những kết quả trên ta thấy:

```
* Dữ liệu không theo phân phối Gaussian
* Dữ liệu không có outlier
=> Dùng MinMaxScaler để chuẩn hóa
```

In [16]:
```
mmScaler = MinMaxScaler()
mmScaler.fit(data[['income', 'age']])
data_sub = mmScaler.transform(data[['income', 'age']])
```

In [17]:
```
data_sub
```

Out[17]:
```
array([[0.56746461, 0.88888889],
       [0.27662387, 0.2       ],
       [0.20367107, 0.75555556],
       ...,
       [0.23538434, 0.37777778],
       [0.0946769 , 0.28888889],
       [0.69080393, 0.57777778]])
```

In [18]:
```
data_sub_min_max_scaler = pd.DataFrame(data_sub,
                                       columns=['mm_income', 'mm_age'])
data = pd.concat([data.reset_index(drop=True),
                  data_sub_min_max_scaler], axis=1)
```

In [19]:
```
data.head()
```

Out[19]:

	income	age	gender_Male	mm_income	mm_age
0	101743	58	0	0.567465	0.888889
1	49597	27	0	0.276624	0.200000
2	36517	52	1	0.203671	0.755556
3	33223	49	1	0.185299	0.688889
4	72994	53	0	0.407119	0.777778

In [20]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
income         1000 non-null int64
age            1000 non-null int64
gender_Male    1000 non-null uint8
mm_income      1000 non-null float64
mm_age         1000 non-null float64
dtypes: float64(2), int64(2), uint8(1)
memory usage: 32.4 KB
```

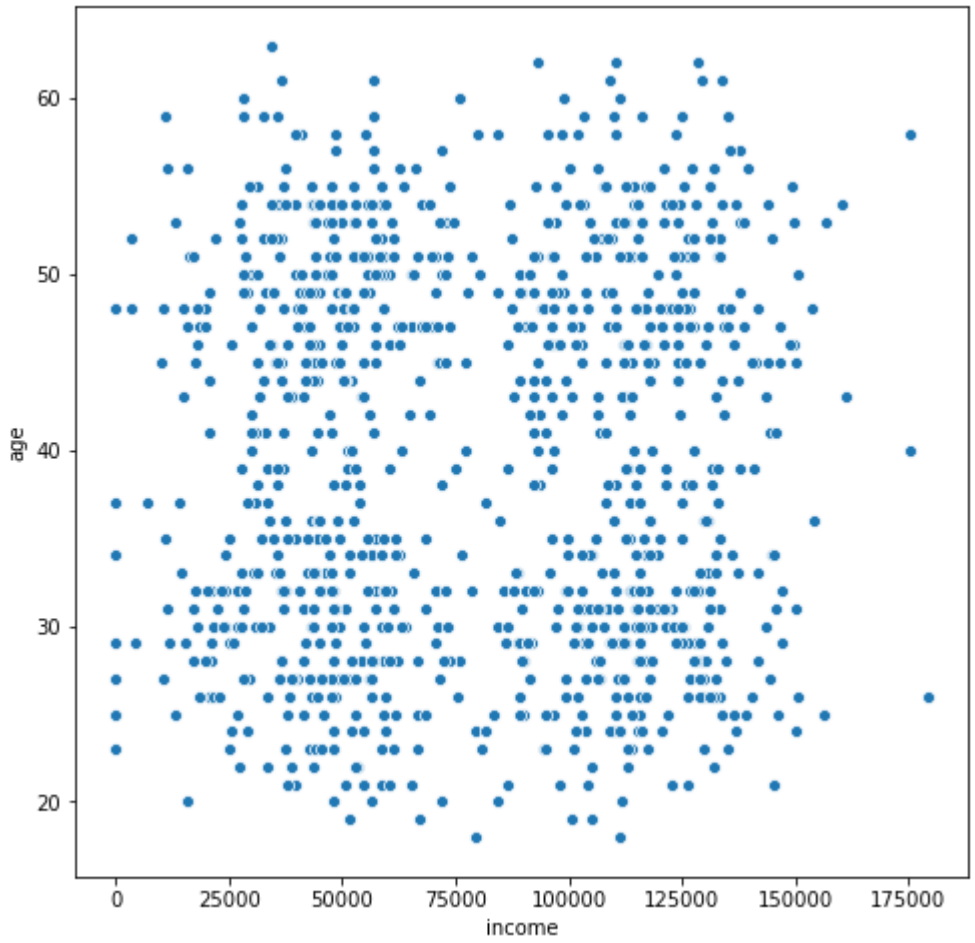
- Data Reduction: lựa chọn các biến phù hợp => đưa vào huấn luyện

```
* Trong bài toán này: sẽ dùng inputs là: mm_income, mm_age [, gender_Mal
e]
```

Step 4: Modeling & Evaluation/ Analyze & Report

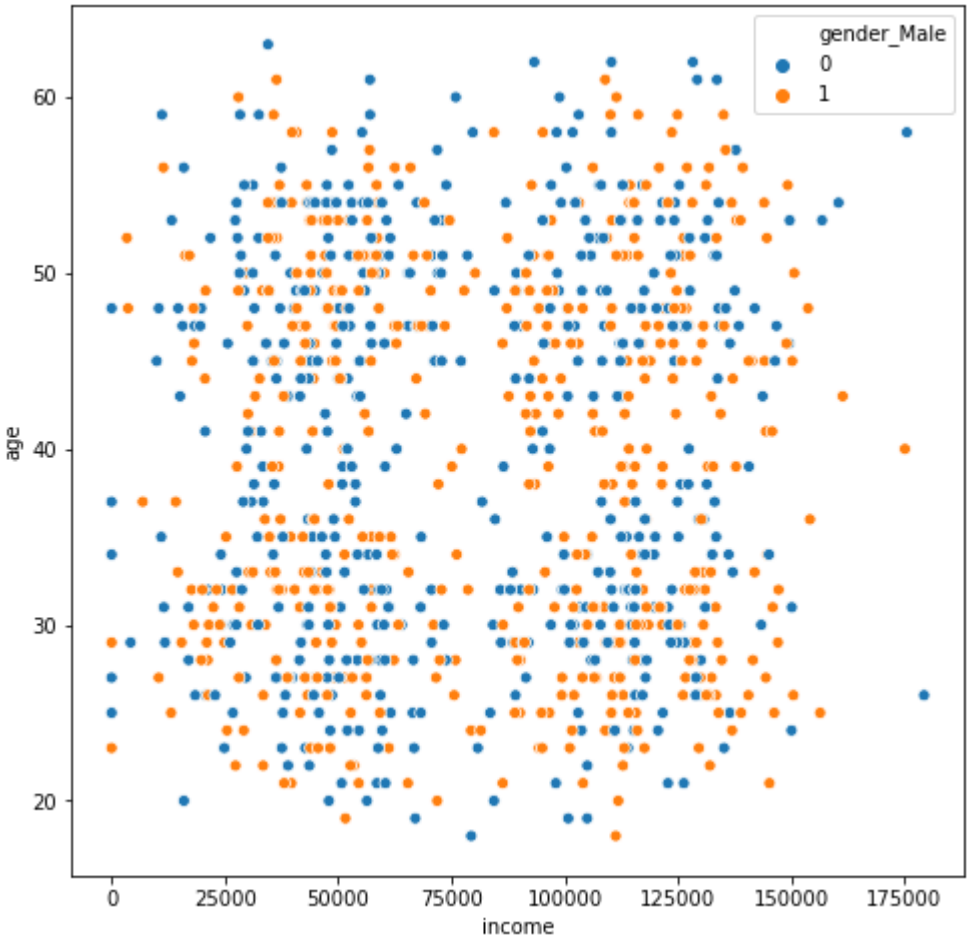
Analyze

In [21]:
```
# Trực quan hóa dữ liệu để có cái nhìn ban đầu về việc phân nhóm
# Income & Age
plt.figure(figsize=(8,8))
sns.scatterplot(data=data, x='income', y='age')
plt.show()
```



In [22]:
```
# Income, Age, Gender
```

In [23]:
```
plt.figure(figsize=(8,8))
sns.scatterplot(data=data, x='income', y='age', hue='gender_Male')
plt.show()
```

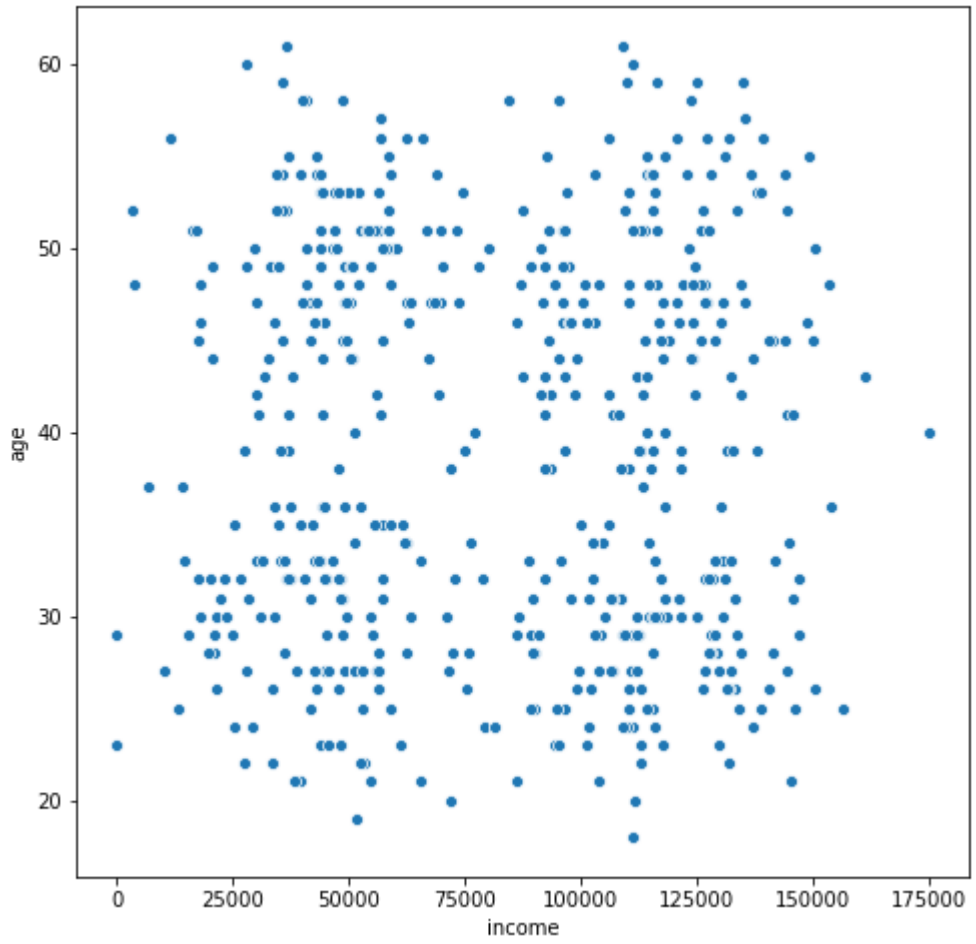


In [24]:
```
# Gender: Male
```

In [25]:
```
data_male = data.where(data['gender_Male']==1, axis=0).dropna()
```

In [26]:
```
# data_male
```

In [27]:
```
plt.figure(figsize=(8,8))
sns.scatterplot(data=data_male, x='income', y='age')
plt.show()
```

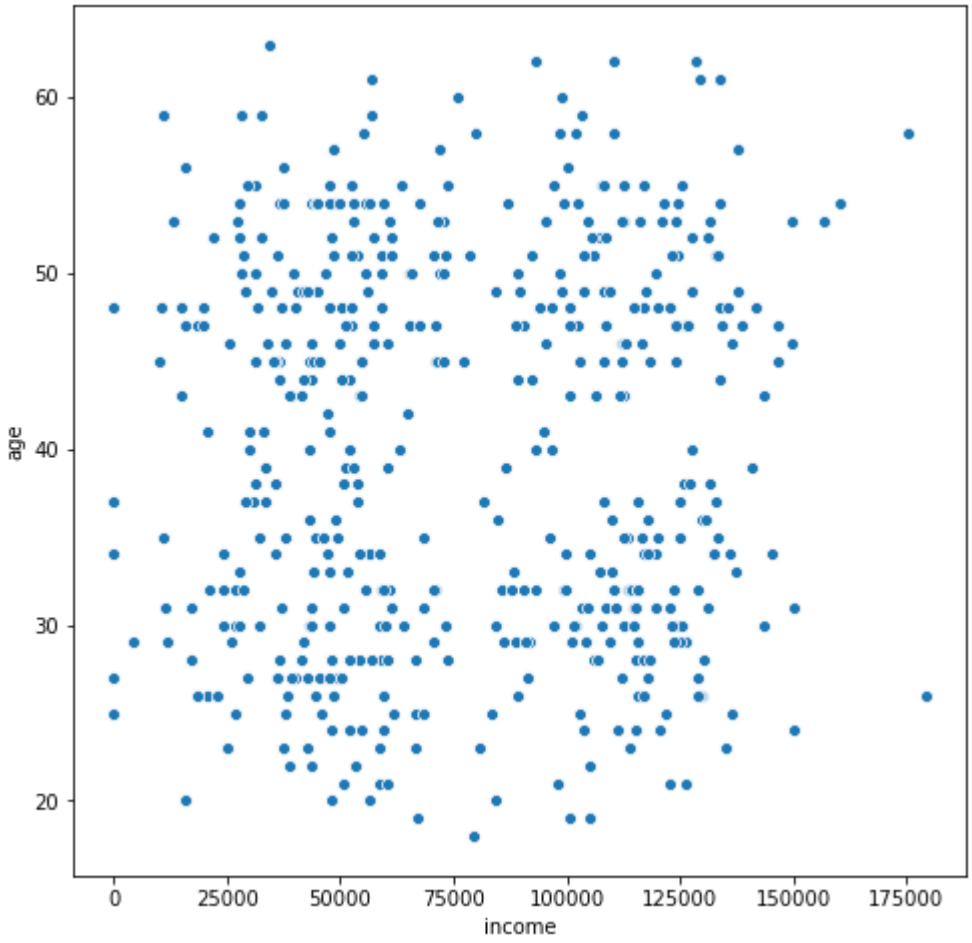


In [28]:
```
# Gender: Female
```

In [29]:
```
data_female = data.where(data['gender_Male']==0, axis=0).dropna()
```

In [30]:
```
# data_female
```

In [31]:
```
plt.figure(figsize=(8,8))
sns.scatterplot(data=data_female, x='income', y='age')
plt.show()
```



* Quan sát các sơ đồ ta thấy dữ liệu có thể phân chia thành 4 cụm.
* Áp dụng thuật toán K-means để phân cụm, có thể kiểm tra lại số cụm (k) bằng phương pháp Elbow method

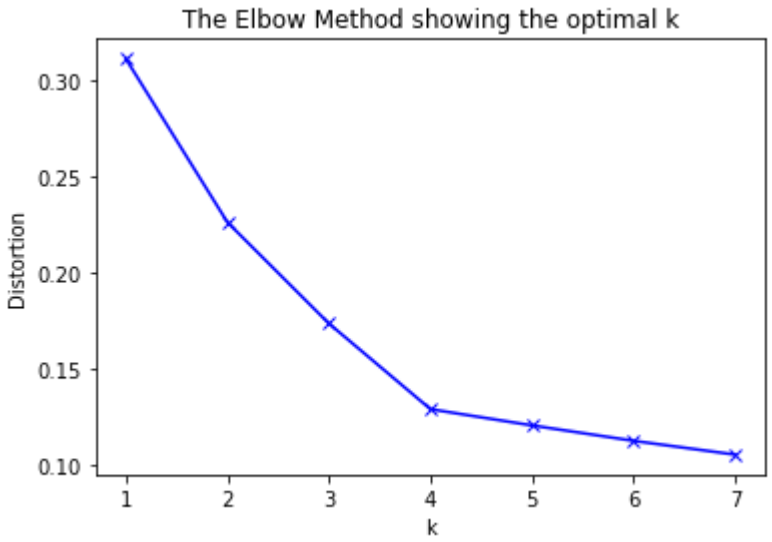
Phân cụm theo: mm_income, mm_age

▶ Gợi ý cách chọn k

In [32]:
```
# Chọn k
distortions = [] # chứa WSSE tương ứng
K = range(1,8)
for k in K:
    kmeanModel = KMeans(n_clusters=k) # số cụm
    kmeanModel.fit(data[['mm_income', 'mm_age']])
    distortions.append(sum(np.min(cdist(data[['mm_income', 'mm_age']],
                                        kmeanModel.cluster_centers_,
                                        'euclidean'), axis=1)) / data[['mm_incom
```

▶ Gợi ý Trực quan elbow

In [33]:
```
# 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 [34]:
```
# Chọn k = 4 vì từ 4 lỗi giảm đi nhỏ và mịn
```

In [35]:
```
# Áp dụng k = 4
kmeans = KMeans(n_clusters=4)
kmeans.fit(data[['mm_income', 'mm_age']])
```

Out[35]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [36]:
```
centroids = kmeans.cluster_centers_
labels = kmeans.labels_
print("Centroids in min-max scale:")
print(centroids)
print("Centroids in normal:")
normal_centroids = mmScaler.inverse_transform(centroids)
print(normal_centroids)
print(labels)
```

```
Centroids in min-max scale:
[[0.64257984 0.26106804]
 [0.25925224 0.69644097]
 [0.64809667 0.6977591 ]
 [0.24372937 0.25833333]]
Centroids in normal:
[[115210.70930233     29.74806202]
 [ 46482.37109375     49.33984375]
 [116199.84453782     49.39915966]
 [ 43699.21370968     29.625     ]]
[2 3 1 1 1 3 3 0 3 1 3 3 3 2 2 1 0 0 3 1 3 0 2 2 3 0 2 3 2 0 3 3 3 0 0 0 3
 3 1 2 1 1 2 0 0 0 2 2 0 3 0 1 2 3 1 3 3 3 0 3 2 1 0 0 3 0 1 0 2 0 1 0 0 2
 1 1 3 0 0 2 2 3 0 1 0 1 1 1 1 0 1 3 1 0 3 3 0 1 1 1 3 0 1 2 1 3 0 1 1 1 3
 2 2 3 0 1 2 0 3 1 2 3 0 3 1 3 0 3 3 2 0 3 1 0 2 1 0 3 2 3 1 3 1 0 3 1 2 0
 0 3 0 3 0 1 3 3 0 3 0 2 2 0 2 0 2 3 3 2 0 3 0 1 0 2 1 1 2 2 0 0 1 3 0 3 2
 0 0 2 2 0 0 1 3 3 3 3 2 0 2 1 2 1 1 3 1 2 2 1 1 1 1 1 2 2 1 3 0 3 2 1 0 0
 3 0 0 2 1 3 1 2 2 1 2 0 3 0 3 2 3 2 2 1 3 0 1 2 2 3 3 0 3 3 3 1 1 1 0 0 2
 0 0 3 2 0 3 2 2 3 0 1 1 0 3 2 2 2 3 3 2 2 2 0 3 2 0 2 1 3 1 3 3 1 2 0 3 0
 1 0 2 3 0 1 0 2 3 1 1 0 3 1 0 0 1 1 1 3 0 3 2 1 1 3 2 1 1 3 2 1 1 1 1 3 0
 0 1 3 3 3 2 2 3 0 0 1 1 3 2 3 1 3 2 3 2 2 0 0 1 0 2 1 2 2 2 1 2 0 3 2 1 0
 1 0 3 1 0 3 1 3 3 3 2 3 3 2 0 0 1 3 2 2 0 3 0 2 2 3 2 2 0 3 1 3 1 1 0 2 1
 0 0 1 3 0 1 1 3 2 0 1 1 1 0 0 1 0 3 1 3 2 2 3 2 1 1 2 3 3 3 1 0 1 2 2 0 2
 1 2 3 0 3 3 1 1 2 3 1 2 2 3 2 3 0 1 0 0 3 0 2 1 3 1 3 3 2 1 1 2 0 0 3 0 0
 0 2 2 3 0 1 1 3 3 3 0 1 1 0 2 1 2 2 0 1 1 0 1 3 0 0 0 1 2 1 2 3 2 2 0 0 1
 2 2 2 1 1 3 2 2 0 2 1 1 1 1 3 3 3 3 0 1 1 1 2 3 2 0 3 3 0 2 1 3 2 0 2 1 3
 0 1 0 3 0 0 3 0 2 2 3 2 0 2 1 3 2 1 2 1 3 3 3 0 3 0 2 0 3 2 2 3 3 2 3 0 0
 3 1 1 3 3 1 1 3 0 1 1 3 0 3 2 2 2 2 0 1 2 1 3 0 2 0 1 2 2 1 3 1 3 3 0 1 2
 0 2 1 0 0 3 0 0 3 0 3 1 3 2 1 0 1 1 1 0 2 1 2 2 0 1 0 1 0 0 3 3 2 1 1 0 1
 2 2 2 1 3 3 3 1 2 0 0 0 1 3 3 0 2 1 3 1 2 1 0 1 3 0 2 0 2 0 3 3 0 3 3 1 0
 0 0 2 0 2 0 3 3 1 0 2 3 1 0 3 2 2 1 2 2 0 3 3 2 2 1 2 0 3 2 0 0 0 1 2 0 2
 0 3 2 3 0 1 1 2 0 2 0 1 3 1 0 0 2 3 0 1 2 0 2 2 0 3 1 3 1 2 2 2 1 0 0 2 1
 1 1 0 2 3 3 1 0 0 2 2 1 2 2 2 0 1 0 0 3 1 1 1 3 2 2 0 1 1 3 2 1 0 0 0 0 2
 2 1 3 0 0 1 3 1 3 3 1 2 0 3 2 0 1 1 2 1 3 2 1 0 0 2 0 1 0 3 1 3 3 1 0 0 2
 1 3 3 3 3 2 0 0 3 3 1 1 3 0 3 0 2 2 0 2 3 3 2 1 1 0 0 2 3 1 3 2 0 1 1 0 3
 3 1 1 2 0 0 3 3 3 1 2 0 1 1 0 2 3 1 1 3 0 2 1 3 1 1 2 0 2 0 3 1 2 0 3 0 0
 2 0 3 0 0 3 3 2 0 0 1 1 0 2 0 0 0 2 3 2 0 2 2 1 1 0 1 3 2 2 0 2 3 2 1 1 2
 3 0 0 0 1 1 2 1 3 2 1 3 1 1 0 0 2 1 2 1 1 1 1 1 1 3 0 3 2 2 2 0 1 3 2 3 3
 2]
```

Report

In [37]:
```
data['Group'] = pd.Series(labels)
data.head()
```

Out[37]:

	income	age	gender_Male	mm_income	mm_age	Group
0	101743	58	0	0.567465	0.888889	2
1	49597	27	0	0.276624	0.200000	3
2	36517	52	1	0.203671	0.755556	1
3	33223	49	1	0.185299	0.688889	1
4	72994	53	0	0.407119	0.777778	1

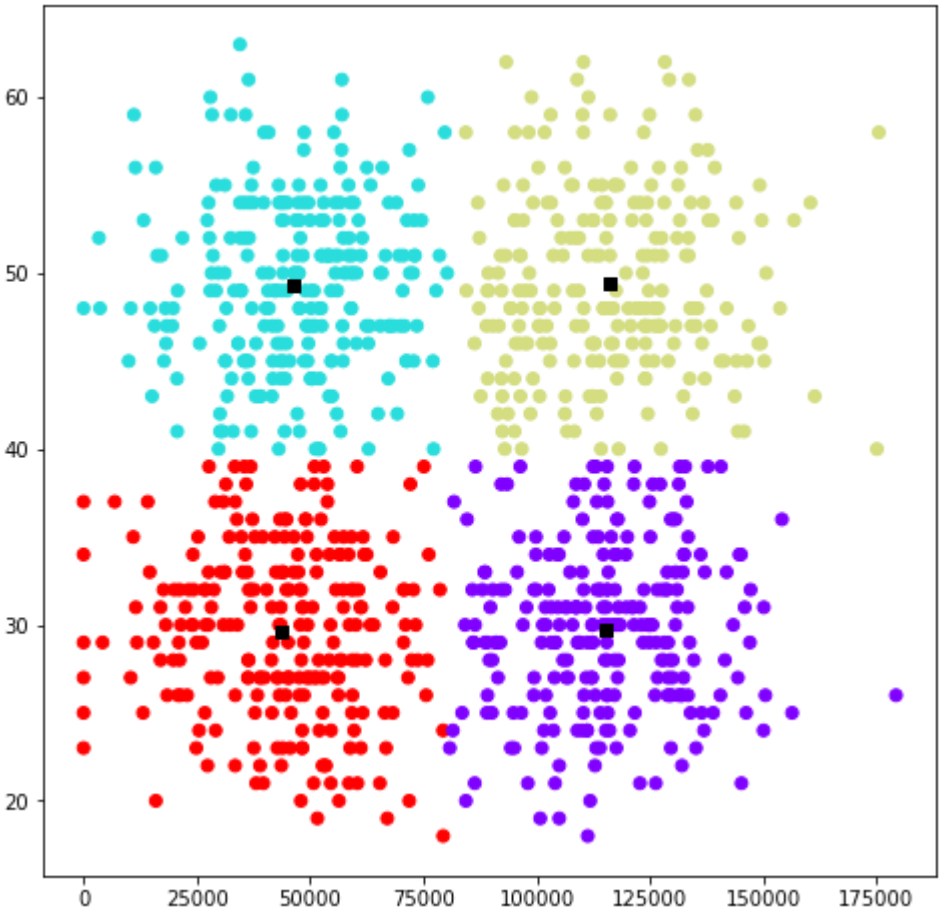
In [38]:
```
data.Group.value_counts()
# Trong đó mỗi nhóm có số khách hàng gần như nhau:
```

Out[38]:
```
0    258
1    256
3    248
2    238
Name: Group, dtype: int64
```

In [39]:
```
i = 0
for cluster in normal_centroids:
    print("*** Centroid Cluster: " + str(i))
    print("Income:",round(cluster[0],2))
    print("Age:",round(cluster[1]),2)
    i = i+1
```

```
*** Centroid Cluster: 0
Income: 115210.71
Age: 30.0 2
*** Centroid Cluster: 1
Income: 46482.37
Age: 49.0 2
*** Centroid Cluster: 2
Income: 116199.84
Age: 49.0 2
*** Centroid Cluster: 3
Income: 43699.21
Age: 30.0 2
```

In [40]:
```
plt.figure(figsize=(8,8))
plt.scatter(data.income, data.age, c= data.Group, cmap='rainbow')
plt.scatter(normal_centroids[:, 0], normal_centroids[:, 1],
            marker = "s",c='black')
plt.show()
```



Có thể chia khách hàng thành 4 nhóm

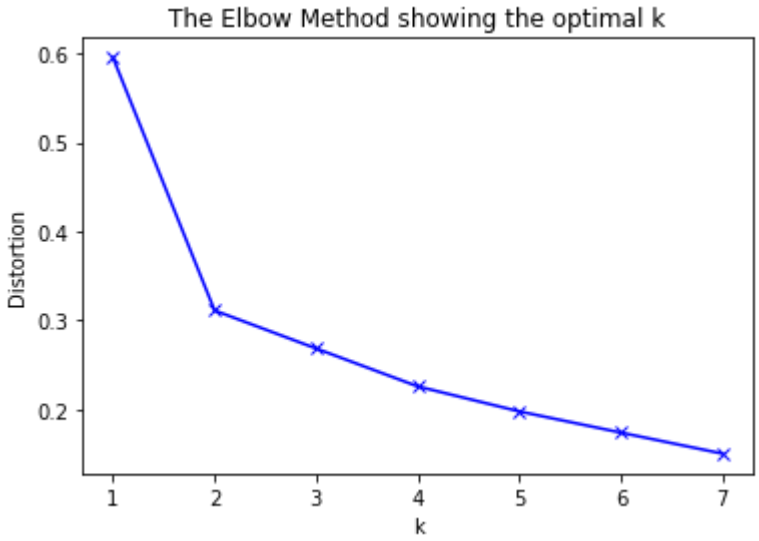
- Nhóm 1: Tuổi <=40 và thu nhập <=75K
- Nhóm 2: Tuổi >40 và thu nhập <=75K
- Nhóm 3: Tuổi <=40 và thu nhập >75K
- Nhóm 4: Tuổi >40 và thu nhập >75K

Phân cụm theo: mm_income, mm_age, Gender

Analyze

In [41]:
```
# Chọn k
distortions = [] # chua WSSE tuong ung
K = range(1,8)
for k in K:
    kmeanModel = KMeans(n_clusters=k) # so cum
    kmeanModel.fit(data[['mm_income', 'mm_age','gender_Male']])
    distortions.append(sum(np.min(cdist(data[['mm_income', 'mm_age', 'gender_Mal
                                        kmeanModel.cluster_centers_,
                                        'euclidean'), axis=1)) / data[['mm_incom
```

In [42]:
```
# 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 [43]:
```
# Áp dụng k = 2 / hoặc k = 4
# Ví dụ: chọn k = 4
kmeans_3 = KMeans(n_clusters=4)
kmeans_3.fit(data[['mm_income', 'mm_age', 'gender_Male']])
```

Out[43]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [44]:
```
centroids_3 = kmeans_3.cluster_centers_
labels_3 = kmeans_3.labels_
print("Centroids in min-max scale:")
print(centroids_3)
print(labels_3)
```

```
Centroids in min-max scale:
[[0.4350457  0.2692548  0.        ]
 [0.45110407 0.25026042 1.        ]
 [0.4705367  0.68313725 1.        ]
 [0.42937575 0.71027778 0.        ]]
[3 0 2 2 3 0 1 0 0 2 0 0 1 3 2 2 0 1 1 3 0 0 3 2 1 0 2 1 2 1 1 1 0 0 1 0 0
 0 2 2 3 2 2 0 1 1 2 2 0 0 1 2 2 0 2 0 1 0 0 0 3 3 0 0 0 0 3 0 3 2 2 1 1 3
 2 2 0 1 0 2 2 0 1 3 1 3 3 2 3 1 3 0 3 0 1 0 1 2 3 2 1 0 3 3 2 1 1 3 3 3 0
 3 2 0 1 2 2 1 1 3 2 1 1 1 2 0 1 1 0 3 0 1 2 0 2 3 1 0 0 0 3 1 2 2 1 2 2 1
 1 1 0 0 0 3 1 0 1 1 0 3 2 1 3 0 3 0 0 2 0 0 1 2 1 2 2 2 3 2 1 0 2 1 0 0 3
 1 1 3 2 1 0 3 1 0 0 1 3 0 2 3 2 2 3 0 2 2 3 3 2 3 3 2 3 2 3 0 1 1 2 3 0 0
 0 0 0 2 3 1 3 2 2 2 2 0 1 1 0 3 1 3 2 2 1 1 3 2 2 1 0 1 1 1 0 3 2 3 0 0 3
 0 1 0 3 1 1 2 3 1 0 2 3 0 1 3 3 3 0 0 3 2 2 0 1 2 1 3 2 1 2 1 1 3 2 0 0 0
 2 1 2 0 1 3 1 2 1 3 3 1 0 2 0 1 2 3 2 1 1 0 3 3 3 0 2 2 3 0 2 2 3 3 2 1 0
 1 2 1 0 0 2 2 1 1 0 2 2 1 2 1 3 0 3 0 3 2 0 1 2 0 2 2 2 2 2 3 3 0 1 2 2 1
 2 1 1 3 0 1 3 0 0 1 3 0 1 3 0 1 3 0 2 2 0 0 1 2 2 0 2 3 1 1 2 1 3 3 0 3 3
 0 1 2 1 0 3 3 1 2 1 3 3 3 0 1 3 0 0 2 0 2 3 0 2 3 3 3 1 1 1 3 1 2 3 2 0 3
 3 3 0 1 0 0 2 3 2 0 2 3 2 0 3 1 0 3 1 2 1 0 2 2 0 3 1 0 3 2 3 2 1 1 1 0 0
 2 2 2 1 1 3 2 0 1 1 1 3 2 1 2 2 2 3 1 3 3 1 3 1 1 1 1 3 3 2 2 1 2 2 0 1 2
 3 2 3 2 2 0 2 3 0 2 3 2 2 2 0 0 1 1 0 3 2 3 3 1 3 1 0 1 0 2 3 1 2 0 3 3 1
 0 2 0 0 1 1 1 1 3 3 1 2 1 3 2 0 2 3 3 3 1 0 1 1 1 0 2 1 0 3 2 0 1 2 0 0 1
 0 2 2 0 0 3 3 0 1 3 3 0 0 0 2 2 3 2 1 3 3 2 1 1 3 1 3 2 3 2 1 3 0 0 0 3 2
 1 3 2 0 0 0 1 1 1 1 1 2 0 2 2 1 3 3 3 0 3 2 3 2 1 2 1 3 0 0 0 1 2 2 3 1 3
 2 3 3 3 0 1 1 3 3 0 1 0 3 0 0 0 3 3 1 3 2 3 1 3 1 0 2 1 3 1 0 0 1 0 1 2 0
 1 2 2 1 3 1 0 0 2 0 2 1 2 0 1 2 2 2 2 2 1 1 0 2 2 2 2 1 0 3 1 0 1 2 2 0 2
 0 0 3 0 1 3 2 2 1 3 0 2 0 3 0 0 2 0 1 3 2 0 3 3 0 1 3 1 3 2 3 3 3 0 0 3 3
 3 3 1 2 1 1 3 1 1 3 3 3 3 2 2 1 3 1 0 0 2 2 2 1 3 2 1 3 3 1 3 2 1 1 1 1 2
 0 3 1 0 0 2 0 2 1 1 2 2 1 0 3 0 3 2 3 2 1 2 2 1 0 3 1 3 0 0 2 0 1 2 1 0 2
 2 1 1 1 0 3 1 1 1 1 2 3 1 1 1 0 2 3 2 2 0 1 2 3 2 2 1 3 0 2 0 3 0 2 2 0 0
 0 3 2 3 0 1 1 0 1 3 3 1 3 2 0 2 1 2 2 1 0 3 3 1 2 3 0 1 3 0 1 3 2 1 1 0 0
 2 0 1 1 0 0 0 3 0 1 2 2 1 3 0 0 0 3 0 3 1 3 3 3 3 0 2 1 3 3 1 2 0 2 3 3 2
 0 0 1 0 3 2 2 3 0 3 3 1 2 2 1 1 2 3 3 2 3 2 3 2 3 1 0 1 3 3 2 2 3 0 3 1 0
 2]
```

Report

In [45]:
```
data['Group_3'] = pd.Series(labels_3)
data.head()
```

Out[45]:

	income	age	gender_Male	mm_income	mm_age	Group	Group_3
0	101743	58	0	0.567465	0.888889	2	3
1	49597	27	0	0.276624	0.200000	3	0
2	36517	52	1	0.203671	0.755556	1	2
3	33223	49	1	0.185299	0.688889	1	2
4	72994	53	0	0.407119	0.777778	1	3

In [46]:
```
data["Group_3"].value_counts()
```

Out[46]:
```
1    256
2    255
0    245
3    244
Name: Group_3, dtype: int64
```

In [47]:
```
for cluster in data['Group_3'].unique():
    print("Cluster: " + str(cluster))
    for col in ['income', 'age']:
        print(col + ": {:.2f}".format(data.loc[data['Group_3'] == cluster, col].m
```

```
Cluster: 3
income: 76583.03
age: 49.80
Cluster: 0
income: 78417.51
age: 29.96
Cluster: 2
income: 84364.41
age: 48.74
Cluster: 1
income: 80880.25
age: 29.26
```

In [48]:
```
from mpl_toolkits.mplot3d import axes3d
```

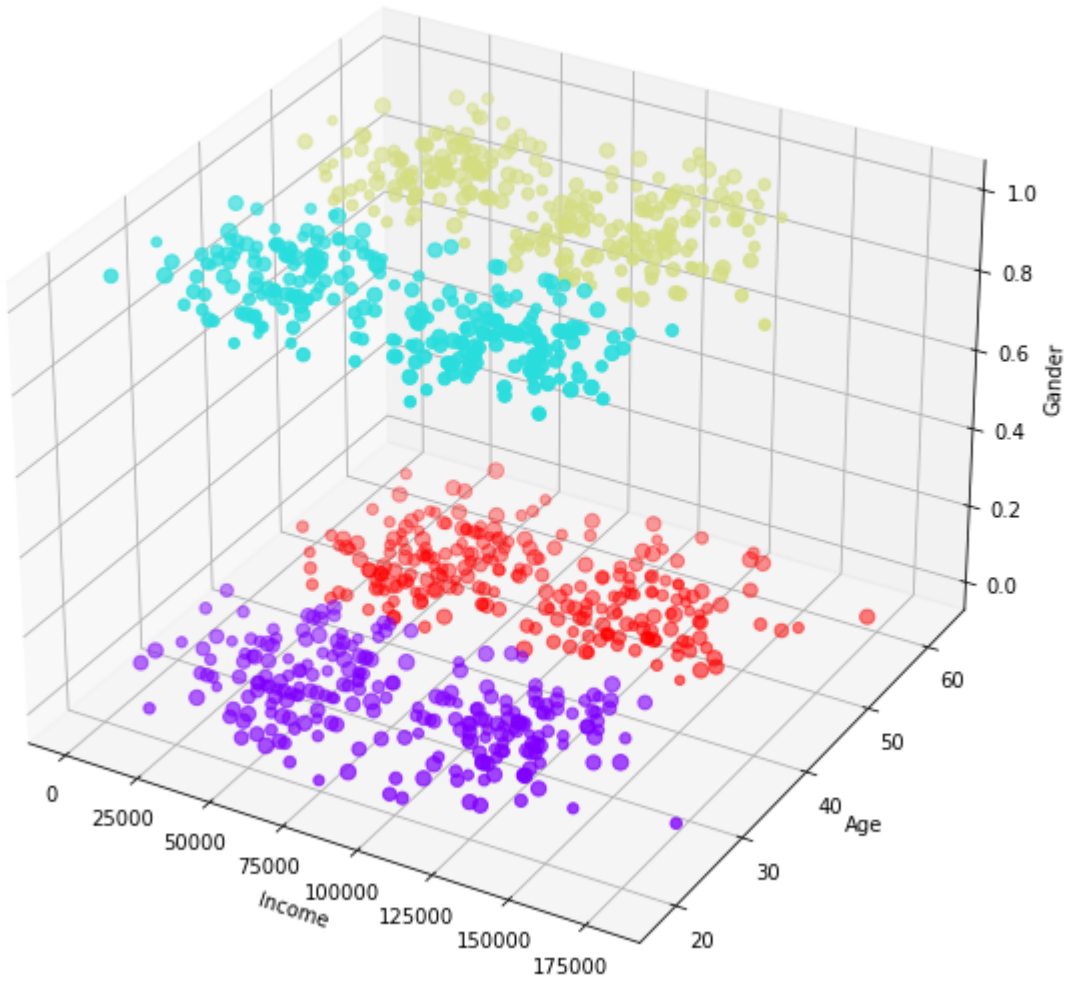
In [49]:
```
fig = plt.figure(figsize=(10,10))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(data.income, data.age, data.gender_Male,
           cmap='rainbow',
           s=data.age,
           c=data.Group_3
           )

ax.set_xlabel('Income')
ax.set_ylabel('Age')
ax.set_zlabel('Gander')
```

Out[49]:
```
Text(0.5, 0, 'Gander')
```



Kết luận: Khi kết hợp 3 yếu tố, ta thấy chỉ có 2 yếu tố ảnh hưởng là Gender và Age, ta có thể chia số nhóm là 4 nhóm như sau:

- Gender = 1 (Male) và Age <= 43
- Gender = 1 (Male) và Age > 43
- Gender = 0 (Female) và Age <=43
- Gender = 0 (Female) và Age >43

Step 5: Deployment & Feedback/ Act

- Đưa ra chiến dịch quảng cáo, bán hàng, chăm sóc khách hàng phù hợp cho mỗi nhóm

In [ ]: