1.

```
print(z_score_normalized)
```

```
       income       age
0    0.550812  1.693570
1   -0.777331 -1.130565
2   -1.110474  1.146963
3   -1.194372  0.873660
4   -0.181416  1.238064
..        ...       ...
995 -0.242008 -0.948363
996  0.381668  0.144851
997 -0.965654 -0.401756
998 -1.608203 -0.766161
999  1.114049  0.418154

[1000 rows x 2 columns]
```

In [123]:

```
data['age_Z']=z_score_normalized['age']
data['income_Z']=z_score_normalized['income']
data.head()
```

Out[123]:

	income	age	age_Z	income_Z
0	101743	58	1.693570	0.550812
1	49597	27	-1.130565	-0.777331
2	36517	52	1.146963	-1.110474
3	33223	49	0.873660	-1.194372
4	72994	53	1.238064	-0.181416

2.(1)

In [124]:

```
from sklearn import cluster
model = cluster.KMeans(n_clusters=2, random_state=10)
X = data[['age_Z', 'income_Z']].values
model.fit_predict(X)
cluster_assignment = model.labels_
centers = model.cluster_centers_
```

In [125]:

```
import numpy as np
print(np.sum((X - centers[cluster_assignment])**2))
```

```
1189.7476232504307
```

In [126]:

```
import matplotlib.pyplot as plt
ss = []
krange = list(range(2, 11))
X = data[['age_Z', 'income_Z']].values
for n in krange:
    model = cluster.KMeans(n_clusters=n, random_state=10)
    model.fit_predict(X)
    cluster_assignments = model.labels_
    centers = model.cluster_centers_
    ss.append(np.sum((X-centers[cluster_assignments])**2))
ss
```

Out[126]:

```
[1189.7476232504307
```

2.(2)

```
In [125]:
import numpy as np
print(np.sum((X - centers[cluster_assignment])**2))
```

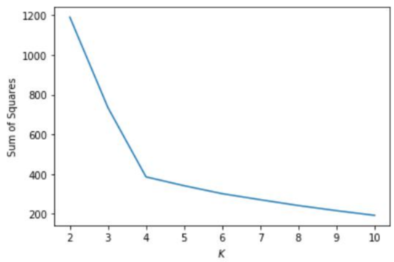
```
1189.7476232504307
```

```
In [126]:
import matplotlib.pyplot as plt
ss = []
krange = list(range(2, 11))
X = data[['age_Z', 'income_Z']].values
for n in krange:
    model = cluster.KMeans(n_clusters=n, random_state=10)
    model.fit_predict(X)
    cluster_assignments = model.labels_
    centers = model.cluster_centers_
    ss.append(np.sum((X-centers[cluster_assignments])**2))
ss
```

```
Out[126]: [1189.7476232504307,
 734.6406648547709,
 386.0702389582008,
 341.6447741546658,
 301.6035411699272,
 271.16618461596397,
 241.90179918136994,
 216.1272992902626,
 192.56868717475388]
```

2.(3)

```
In [127]:
plt.plot(krange, ss)
plt.xlabel('$K$')
plt.ylabel('Sum of Squares')
plt.show()
#4為最佳分群數
```




```
In [129]:
from sklearn import cluster
model = cluster.KMeans(n_clusters=4, random_state=10)
X = data[['age_Z', 'income_Z']].values
model.fit_predict(X)
cluster_assignment = model.labels_
```

3.

```
In [129]:
from sklearn import cluster
model = cluster.KMeans(n_clusters=4, random_state=10)
X = data[['age_Z', 'income_Z']].values
model.fit_predict(X)
cluster_assignment = model.labels_
centers = model.cluster_centers_
data['kmeans_cluster']=cluster_assignment
data.head()
```

Out[129]:

	income	age	age_Z	income_Z	kmeans_cluster
0	101743	58	1.693570	0.550812	0
1	49597	27	-1.130565	-0.777331	1
2	36517	52	1.146963	-1.110474	2
3	33223	49	0.873660	-1.194372	2
4	72994	53	1.238064	-0.181416	2

4.

```
std = data.std()
#標準化後之結果
z_score_normalized = (data - mu) / std

print(z_score_normalized)
```

```
       income       age
0    0.550812  1.693570
1   -0.777331 -1.130565
2   -1.110474  1.146963
3   -1.194372  0.873660
4   -0.181416  1.238064
..        ...       ...
995 -0.242008 -0.948363
996  0.381668  0.144851
997 -0.965654 -0.401756
998 -1.608203 -0.766161
999  1.114049  0.418154

[1000 rows x 2 columns]
```

In [165]:

```
data['age_Z']=z_score_normalized['age']
data['income_Z']=z_score_normalized['income']
data.head()
```

Out[165]:

	income	age	age_Z	income_Z
0	101743	58	1.693570	0.550812
1	49597	27	-1.130565	-0.777331
2	36517	52	1.146963	-1.110474

5、6

In [166]:

```
from sklearn.cluster import MeanShift, estimate_bandwidth
bandwidth = estimate_bandwidth(data, quantile=0.1)
ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms.fit(data)
labels = ms.labels_
cluster_centers = ms.cluster_centers_
labels_unique = np.unique(labels)
n_clusters_ = len(labels_unique)

print("number of estimated clusters : %d" % n_clusters_)
import matplotlib.pyplot as plt
from itertools import cycle
print(bandwidth)
plt.figure(1)
plt.clf()
data['ms']=labels
data.head()

```

```
number of estimated clusters : 8
7365.635129383834
```

Out[166]:

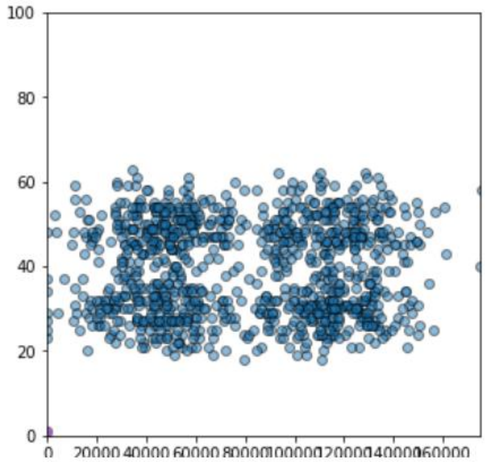
	income	age	age_Z	income_Z	ms
0	101743	58	1.693570	0.550812	4
1	49597	27	-1.130565	-0.777331	0
2	36517	52	1.146963	-1.110474	3
3	33223	49	0.873660	-1.194372	3
4	72994	53	1.238064	-0.181416	5

7

In [181]:

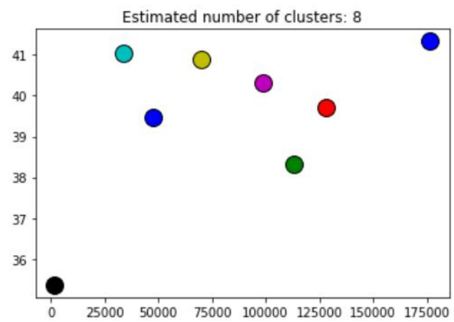
```python
fig =plt.figure(figsize=(5,5))

plt.scatter(data['income'],data['age'],alpha=0.5,edgecolor='k')
for idx, centers in enumerate(centers):
    plt.scatter(*centers)
plt.xlim(0,175000)
plt.ylim(0,100)
plt.show
```

Out[181]: <function matplotlib.pyplot.show(*args, **kw)>



In [100]:

```python
colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
for k, col in zip(range(n_clusters_), colors):
    ##根据lables中的值是否等于k，重新组成一个True、False的数组
    my_members = labels == k
    cluster_center = cluster_centers[k]
    plt.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,
             markeredgecolor='k', markersize=14)
plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```



8.

In [134]:

```python
mu = data['age'].mean()
#標準差
std = data['age'].std()
#標準化後之結果
z_score_normalized = (data['age'] - mu) / std

print(z_score_normalized)
```

```
0     -1.446258
1      0.438698
2      0.783290
3      1.105599
4     -1.381068
         ...
995    1.347424
996   -1.746302
997    0.915534
998    1.060287
999    0.517937
Name: age, Length: 1000, dtype: float64
```

In [135]:

```python
data['age_Z']=z_score_normalized
data.head()
```

Out[135]:

	age	education	age_Z
0	27.007219	college	-1.446258
1	47.615409	highschool	0.438698
2	51.382815	highschool	0.783290

9

```python
label = kp.labels_

```

In [154]:

```python
data['label']=label
data.head()

```

Out[154]:

	age	education	age_Z	label
0	27.007219	college	-1.446258	0
1	47.615409	highschool	0.438698	2
2	51.382815	highschool	0.783290	1
3	54.906622	highschool	1.105599	1
4	27.719939	less_than_highschool	-1.381068	0

In [155]:

```python
sector=data.groupby('label')
sector.size()
```

Out[155]:

```
label
0    316
1    396
2    288
dtype: int64
```

10

Out[155]:

```
label
0    316
1    396
2    288
dtype: int64
```


In [157]:

```python
import pandas as pd
import numpy as np


data=pd.read_csv('C:/Users/Admin/Downloads/customer_offers.csv',index_col="customer_name")
data.head()
```

Out[157]:

customer_name	1	2	3	4	5	6	7	8	9	10	...	23	24	25	26	27	28	29	30	31	32
Adams	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	1	0	0
Allen	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	1	0	0	0	0	0
Anderson	0	0	0	0	0	0	0	0	0	0	...	0	1	0	1	0	0	0	0	0	0
Bailey	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	1	0	0
Baker	0	0	0	0	0	0	1	0	0	1	...	0	0	0	0	0	0	0	0	1	0

5 rows × 32 columns

11

```
In [161]:
from kmodes import kmodes
km = kmodes.KModes(n_clusters=4,random_state=10)
clusters = km.fit_predict(data)
data['km']=clusters
data.head()
```

Out[161]:

customer_name	1	2	3	4	5	6	7	8	9	10	...	24	25	26	27	28	29	30	31	32	km
Adams	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	1	0	0	0
Allen	0	0	0	0	0	0	0	0	1	0	...	0	0	0	1	0	0	0	0	0	0
Anderson	0	0	0	0	0	0	0	0	0	0	...	1	0	1	0	0	0	0	0	0	0
Bailey	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	1	0	0	0
Baker	0	0	0	0	0	0	1	0	0	1	...	0	0	0	0	0	0	0	1	0	0

5 rows × 33 columns

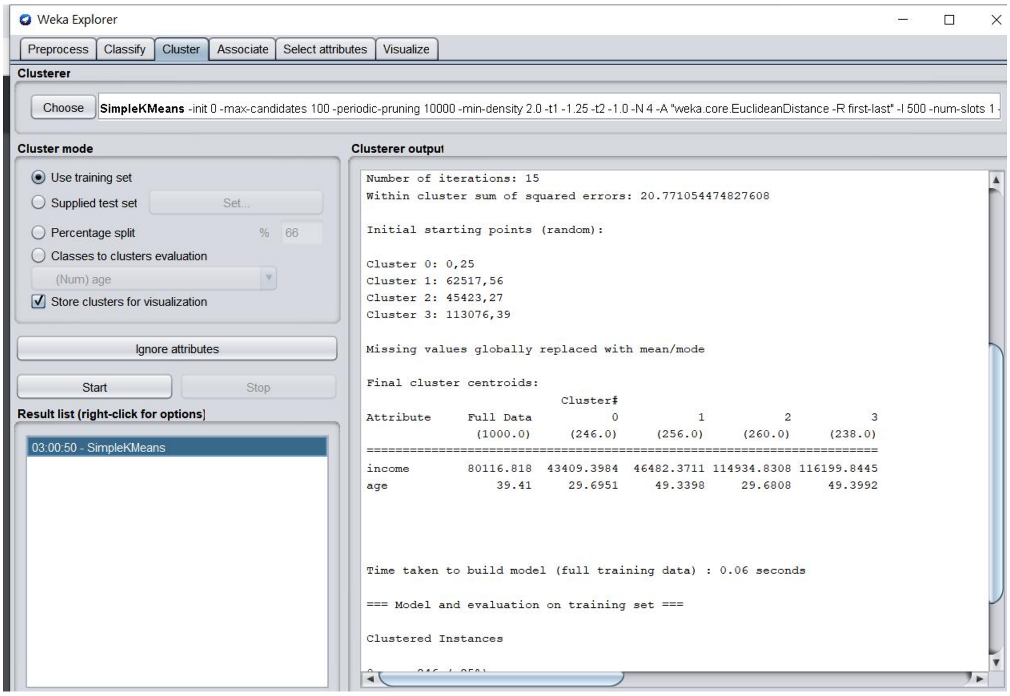
```
In [162]:
sector=data.groupby('km')
sector.size()
```

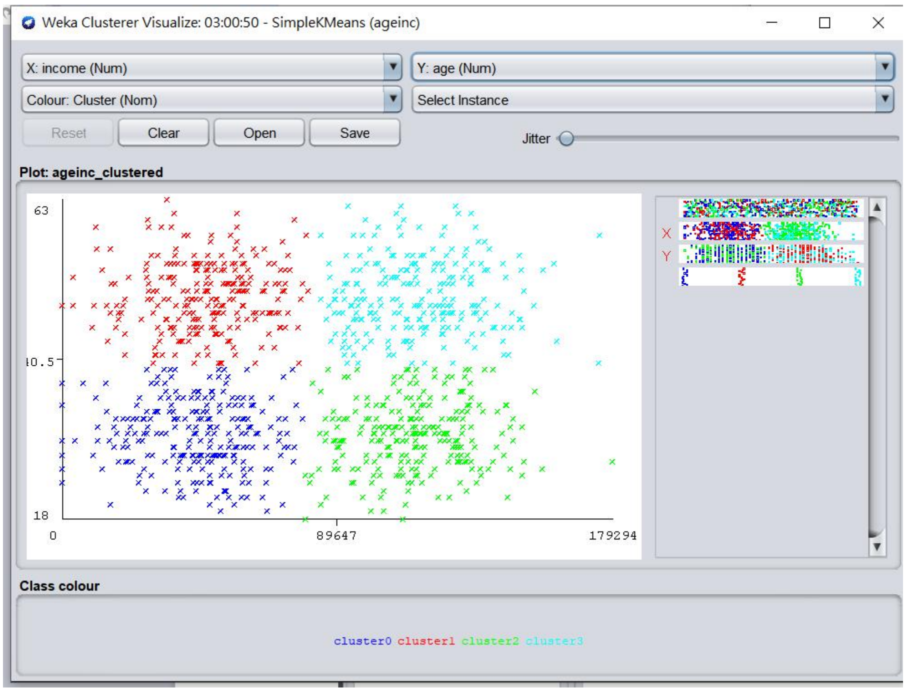
```
Out[162]:
km
0    89
1     2
2     6
3     3
dtype: int64
```

12

```
In [ ]:
k means :適用於數值資料，須定義K，一種無間督學習的算法，演算法可以非常快速地完成分群任務，
    但是如果觀測值具有雜訊（Noise）或者極端值，其分群結果容易被這些雜訊與極端值影響，適合處理分布集中的大型樣本資料。

Mean Shift :適用於數值資料，不需要預先知道欲分群的數目，
K-prototype:同時有數值或離散可使用，是K-means與K-modes的一種集合形式，透過設定了一個目標函數，透過不斷迭代運算，直到目標函數值不變
K-modes:適用於類別型，搜尋群聚中心的方法是以各類別資料的眾數（Modes）為依據，，將原本K-means使用的歐式距離替換成字符間的漢明距離
```

13




Weka Clusterer Visualize: 03:00:50 - SimpleKMeans (ageinc)
X: income (Num)
Y: age (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: ageinc_clustered
63
40.5
18
0
89647
179294
X
Y
Class colour
cluster0 cluster1 cluster2 cluster3