In [244]:
```
import numpy as np
import pandas as pd
```

In [245]:
```
# generate our own dataset by using make_blobs function from scikit learn module
from sklearn.datasets import make_blobs

print(make_blobs)   # make_blobs function generate blobs of points with gaussioar
#where maximum data falls near mean of the data
```

```
<function make_blobs at 0x0000026290BF6E50>
```

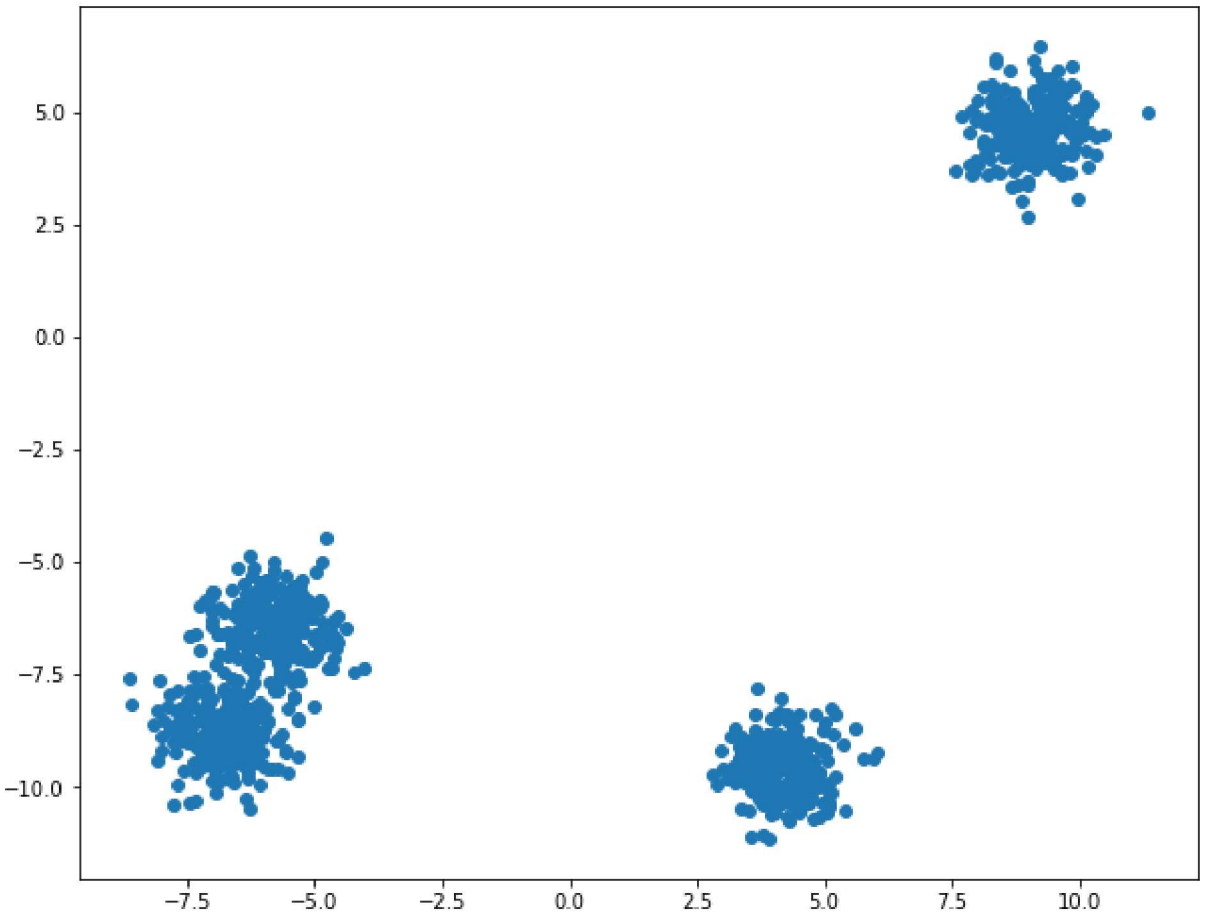
In [246]:
```
X,y =make_blobs(n_samples=1000,centers=4,cluster_std=0.6,n_features=4,random_stat

# cluster_std is standard deviation of clusters
# no.of centers to generate   # no.of groups/clustres
# no.of features for each sample   #no.of columns
```

In [247]:
```
print(X)
```

```
[[ -7.68790296  -7.46307802  -8.11807941   6.92938635]
 [ -6.20515956  -6.31159372  -9.30205436   7.56775429]
 [ -2.78058141   7.55995853   3.68953714   2.42941848]
 ...
 [  0.69518309   5.08151178 -10.44268441   9.23820565]
 [  6.8421022   -5.75987299  -6.85130928  -5.83794185]
 [ -2.83305343   8.54730329   4.75738599   1.38614602]]
```

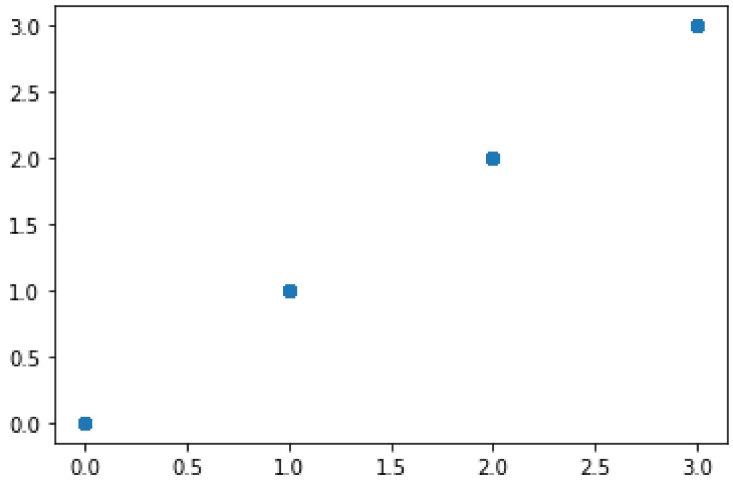
In [248]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,8))
plt.scatter(X[:,1],X[:,2])
```

Out[248]: <matplotlib.collections.PathCollection at 0x26295e148e0>



In [249]:
```
plt.scatter(y[:],y[:])
```

Out[249]:
```
<matplotlib.collections.PathCollection at 0x262970c15b0>
```



In [250]:
```
print(X.shape)
```

```
(1000, 4)
```

In [251]:
```
print(y.shape)
```

```
(1000,)
```

In [252]:
```
print(y)
```

```
[1 1 0 0 1 0 3 1 2 2 0 0 0 3 2 3 1 0 3 0 0 1 0 0 1 2 1 2 3 2 2 3 2 3 3 2 2
 0 2 2 0 3 2 3 1 1 1 3 1 2 3 2 2 2 1 3 1 1 0 2 2 2 1 0 2 2 2 1 3 1 0 0 3 2
 0 0 2 3 1 2 3 1 2 0 0 0 2 1 1 1 1 2 3 3 0 1 0 3 1 0 3 3 1 2 0 3 0 3 1 3 1
 2 3 3 1 2 1 2 2 2 2 0 1 0 1 0 2 2 0 0 3 1 3 2 1 2 1 0 2 3 3 2 1 0 2 3 1 1
 0 3 3 2 3 3 0 2 3 0 3 2 2 0 1 2 3 1 0 2 2 1 2 1 0 2 2 1 3 2 0 2 3 2 3 0 2
 2 2 1 0 0 0 2 1 1 2 0 1 0 3 2 0 3 3 2 1 3 1 0 2 3 0 3 2 3 3 2 3 1 3 0 2 0
 2 1 2 2 2 1 1 3 2 3 2 0 1 1 3 0 1 3 2 3 2 2 2 1 0 0 3 0 1 3 3 2 2 1 2 1 0
 3 0 3 1 2 0 2 3 0 1 0 1 2 3 2 2 0 3 1 3 0 0 0 1 0 0 0 2 0 1 1 1 1 1 0 3 0
 3 0 0 0 1 1 2 0 0 2 1 1 3 1 0 1 1 1 2 2 3 0 3 0 0 3 3 0 0 2 3 3 0 3 0 3 1
 0 1 1 1 3 2 2 1 3 1 2 2 0 3 2 0 1 2 0 0 2 3 0 1 1 3 3 3 3 0 3 0 1 2 3 3 0
 2 1 3 0 3 2 2 1 3 3 3 1 1 0 0 2 3 1 3 3 3 0 2 1 3 1 1 3 0 2 2 2 0 2 3 2 0
 2 0 2 1 1 0 3 0 0 3 1 3 0 3 2 1 2 0 0 1 1 1 2 1 0 1 3 1 2 1 1 3 0 1 0 2 1
 0 2 2 3 2 0 0 1 2 3 0 2 2 1 1 2 2 0 0 1 0 3 1 1 2 0 1 3 3 2 1 3 2 2 1 3 2
 1 2 2 3 0 2 3 3 1 1 3 3 0 2 2 2 0 1 2 1 3 2 3 1 3 3 0 0 3 1 0 0 0 0 3 1 3
 2 1 2 0 3 0 3 3 1 3 2 3 3 0 1 1 0 3 3 1 1 2 0 2 2 0 3 2 0 3 0 2 0 0 1 0 2
 3 3 1 0 3 2 1 1 1 1 1 1 0 3 1 3 3 2 0 0 3 2 1 0 3 1 3 1 2 2 0 3 1 1 2 2 2
 2 3 1 0 3 1 3 2 0 2 0 3 1 2 0 2 1 1 1 0 3 0 0 3 2 2 0 1 2 0 1 1 1 3 2 1 2
 2 3 3 2 3 0 3 1 1 0 2 0 0 2 3 3 1 0 1 3 1 2 3 3 2 2 1 1 0 2 0 1 1 1 2 0 3
 2 0 0 0 0 2 0 1 1 0 1 1 3 1 1 0 1 3 0 1 3 2 2 0 1 0 3 3 0 1 2 0 0 3 0 3 1
 1 0 0 0 2 3 3 3 0 0 3 1 0 2 1 1 3 2 2 0 1 0 2 3 1 1 1 2 3 2 2 0 1 1 2 2 3
 0 2 0 1 0 1 0 2 1 3 3 0 0 1 0 0 1 1 2 3 3 0 2 3 3 2 0 2 3 0 2 3 3 2 0 0 1
 1 3 1 3 3 3 2 0 2 2 3 2 3 3 1 2 0 1 0 1 2 3 3 2 3 3 1 2 3 1 2 2 1 3 0 0 0
 2 0 2 0 1 3 0 2 2 1 1 2 2 3 1 0 0 1 1 2 3 2 2 2 1 1 1 2 1 1 3 2 1 3 0 3 0
 0 2 3 1 1 0 3 0 1 3 0 2 1 0 2 0 2 2 3 2 3 0 3 1 1 0 3 3 3 1 2 0 1 2 0 3 3
 1 1 2 1 3 3 1 0 1 0 0 0 3 1 2 3 1 2 2 2 0 3 2 1 0 2 3 2 0 1 3 1 0 0 1 1 0
 0 3 3 3 3 2 3 0 3 2 1 1 1 1 3 2 0 3 0 3 2 3 2 3 0 0 0 0 0 0 0 1 3 3 1 2 2
 3 2 0 3 2 1 3 3 1 3 2 0 0 3 2 2 1 0 1 3 1 2 3 1 1 3 0 3 0 2 3 3 0 3 2 2 3
 0]
```

In [253]:
```
#y.reshape(1000,1)
```

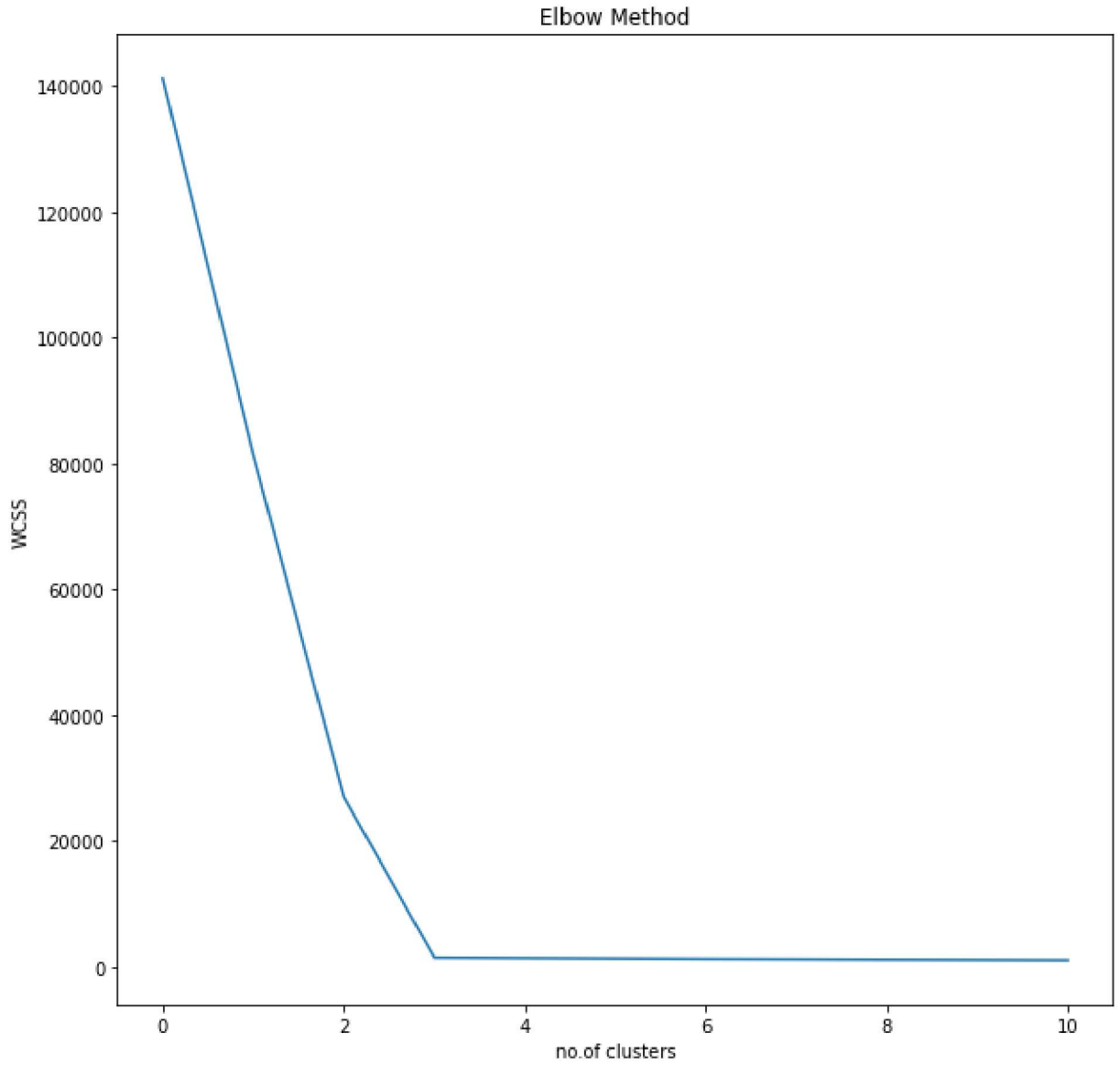
In [254]:
```
print(y.shape)
```

```
(1000,)
```

In [255]:
```
from sklearn.cluster import KMeans
WCSS=[]  # within cluster sum of sqaures
for i in range(1,12):
    kmeans=KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_sta
    print(kmeans)
    kmeans.fit(X)
    WCSS.append(kmeans.inertia_)
print(WCSS)
```

```
KMeans(n_clusters=1, random_state=42)
KMeans(n_clusters=2, random_state=42)

C:\Users\prasa\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=4.
  warnings.warn(

KMeans(n_clusters=3, random_state=42)
KMeans(n_clusters=4, random_state=42)
KMeans(n_clusters=5, random_state=42)
KMeans(n_clusters=6, random_state=42)
KMeans(n_clusters=7, random_state=42)
KMeans(random_state=42)
KMeans(n_clusters=9, random_state=42)
KMeans(n_clusters=10, random_state=42)
KMeans(n_clusters=11, random_state=42)
[141210.88490196608, 81376.7859919679, 26977.79322280449, 1428.7810986483944, 1
357.4573203843224, 1287.9648679133043, 1219.2247643173223, 1174.922421562302, 1
115.8759517935496, 1073.5834362541711, 1027.0298461075124]
```

In [256]:
```
plt.figure(figsize=(10,10))
plt.plot(WCSS)  #if you provide only one parameter, matplotlib assumes it as a y
plt.title("Elbow Method")
plt.xlabel("no.of clusters")
plt.ylabel("WCSS")
plt.show()
```



In [271]:
```
# BY using elbow method, we get the no.of clusters for our data
#so no.of clusters =3
kmeans=KMeans(n_clusters=4,max_iter=300,n_init=10,random_state=42)
pred_y=kmeans.fit_predict(X)
pred_y
```

Out[271]:
```
array([0, 0, 1, 1, 0, 1, 2, 0, 3, 3, 1, 1, 1, 2, 3, 2, 0, 1, 2, 1, 1, 0,
       1, 1, 0, 3, 0, 3, 2, 3, 3, 2, 3, 2, 2, 3, 3, 1, 3, 3, 1, 2, 3, 2,
       0, 0, 0, 2, 0, 3, 2, 3, 3, 3, 0, 2, 0, 0, 1, 3, 3, 3, 0, 1, 3, 3,
       3, 0, 2, 0, 1, 1, 2, 3, 1, 1, 3, 2, 0, 3, 2, 0, 3, 1, 1, 1, 3, 0,
       0, 0, 0, 3, 2, 2, 1, 0, 1, 2, 0, 1, 2, 2, 0, 3, 1, 2, 1, 2, 0, 2,
       0, 3, 2, 2, 0, 3, 0, 3, 3, 3, 3, 1, 0, 1, 0, 1, 3, 3, 1, 1, 2, 0,
       2, 3, 0, 3, 0, 1, 3, 2, 2, 3, 0, 1, 3, 2, 0, 0, 1, 2, 2, 3, 2, 2,
       1, 3, 2, 1, 2, 3, 3, 1, 0, 3, 2, 0, 1, 3, 3, 0, 3, 0, 1, 3, 3, 0,
       2, 3, 1, 3, 2, 3, 2, 1, 3, 3, 3, 0, 1, 1, 1, 3, 0, 0, 3, 1, 0, 1,
       2, 3, 1, 2, 2, 3, 0, 2, 0, 1, 3, 2, 1, 2, 3, 2, 2, 3, 2, 0, 2, 1,
       3, 1, 3, 0, 3, 3, 3, 0, 0, 2, 3, 2, 3, 1, 0, 0, 2, 1, 0, 2, 3, 2,
       3, 3, 3, 0, 1, 1, 2, 1, 0, 2, 2, 3, 3, 0, 3, 0, 1, 2, 1, 2, 0, 3,
       1, 3, 2, 1, 0, 1, 0, 3, 2, 3, 3, 1, 2, 0, 2, 1, 1, 1, 0, 1, 1, 1,
       3, 1, 0, 0, 0, 0, 0, 1, 2, 1, 2, 1, 1, 1, 0, 0, 3, 1, 1, 3, 0, 0,
       2, 0, 1, 0, 0, 0, 3, 3, 2, 1, 2, 1, 1, 2, 2, 1, 1, 3, 2, 2, 1, 2,
       1, 2, 0, 1, 0, 0, 0, 2, 3, 3, 0, 2, 0, 3, 3, 1, 2, 3, 1, 0, 3, 1,
       1, 3, 2, 1, 0, 0, 2, 2, 2, 2, 1, 2, 1, 0, 3, 2, 2, 1, 3, 0, 2, 1,
       2, 3, 3, 0, 2, 2, 2, 0, 0, 1, 1, 3, 2, 0, 2, 2, 2, 1, 3, 0, 2, 0,
       0, 2, 1, 3, 3, 3, 1, 3, 2, 3, 1, 3, 1, 3, 0, 0, 1, 2, 1, 1, 2, 0,
       2, 1, 2, 3, 0, 3, 1, 1, 0, 0, 0, 3, 0, 1, 0, 2, 0, 3, 0, 0, 2, 1,
       0, 1, 3, 0, 1, 3, 3, 2, 3, 1, 1, 0, 3, 2, 1, 3, 3, 0, 0, 3, 3, 1,
       1, 0, 1, 2, 0, 0, 3, 1, 0, 2, 2, 3, 0, 2, 3, 3, 0, 2, 3, 0, 3, 3,
       2, 1, 3, 2, 2, 0, 0, 2, 2, 1, 3, 3, 3, 1, 0, 3, 0, 2, 3, 2, 0, 2,
       2, 1, 1, 2, 0, 1, 1, 1, 1, 2, 0, 2, 3, 0, 3, 1, 2, 1, 2, 2, 0, 2,
       3, 2, 2, 1, 0, 0, 1, 2, 2, 0, 0, 3, 1, 3, 3, 1, 2, 3, 1, 2, 1, 3,
       1, 1, 0, 1, 3, 2, 2, 0, 1, 2, 3, 0, 0, 0, 0, 0, 0, 1, 2, 0, 2, 2,
       3, 1, 1, 2, 3, 0, 1, 2, 0, 2, 0, 3, 3, 1, 2, 0, 0, 3, 3, 3, 3, 2,
       0, 1, 2, 0, 2, 3, 1, 3, 1, 2, 0, 3, 1, 3, 0, 0, 0, 1, 2, 1, 1, 2,
       3, 3, 1, 0, 3, 1, 0, 0, 0, 2, 3, 0, 3, 3, 2, 2, 3, 2, 1, 2, 0, 0,
       1, 3, 1, 1, 3, 2, 2, 0, 1, 0, 2, 0, 3, 2, 2, 3, 3, 0, 0, 1, 3, 1,
       0, 0, 0, 3, 1, 2, 3, 1, 1, 1, 1, 3, 1, 0, 0, 1, 0, 0, 2, 0, 0, 1,
       0, 2, 1, 0, 2, 3, 3, 1, 0, 1, 2, 2, 1, 0, 3, 1, 1, 2, 1, 2, 0, 0,
       1, 1, 1, 3, 2, 2, 2, 1, 1, 2, 0, 1, 3, 0, 0, 2, 3, 3, 1, 0, 1, 3,
       2, 0, 0, 0, 3, 2, 3, 3, 1, 0, 0, 3, 3, 2, 1, 3, 1, 0, 1, 0, 1, 3,
       0, 2, 2, 1, 1, 0, 1, 1, 0, 0, 3, 2, 2, 1, 3, 2, 2, 3, 1, 3, 2, 1,
       3, 2, 2, 3, 1, 1, 0, 0, 2, 0, 2, 2, 2, 3, 1, 3, 3, 2, 3, 2, 2, 0,
       3, 1, 0, 1, 0, 3, 2, 2, 3, 2, 2, 0, 3, 2, 0, 3, 3, 0, 2, 1, 1, 1,
       3, 1, 3, 1, 0, 2, 1, 3, 3, 0, 0, 3, 3, 2, 0, 1, 1, 0, 0, 3, 2, 3,
       3, 3, 0, 0, 0, 3, 0, 0, 2, 3, 0, 2, 1, 2, 1, 1, 3, 2, 0, 0, 1, 2,
       1, 0, 2, 1, 3, 0, 1, 3, 1, 3, 3, 2, 3, 2, 1, 2, 0, 0, 1, 2, 2, 2,
       0, 3, 1, 0, 3, 1, 2, 2, 0, 0, 3, 0, 2, 2, 0, 1, 0, 1, 1, 1, 2, 0,
       3, 2, 0, 3, 3, 3, 1, 2, 3, 0, 1, 3, 2, 3, 1, 0, 2, 0, 1, 1, 0, 0,
       1, 1, 2, 2, 2, 2, 3, 2, 1, 2, 3, 0, 0, 0, 0, 2, 3, 1, 2, 1, 2, 3,
       2, 3, 2, 1, 1, 1, 1, 1, 1, 1, 0, 2, 2, 0, 3, 3, 2, 3, 1, 2, 3, 0,
       2, 2, 0, 2, 3, 1, 1, 2, 3, 3, 0, 1, 0, 2, 0, 3, 2, 0, 0, 2, 1, 2,
       1, 3, 2, 2, 1, 2, 3, 3, 2, 1])
```

In [272]:
```
a=pd.DataFrame(pred_y,columns=["cluster"])
#a["new"]=0
a
```

Out[272]:

	cluster
0	0
1	0
2	1
3	1
4	0
...	...
995	2
996	3
997	3
998	2
999	1

1000 rows × 1 columns

In [273]:
```
b=a.groupby(["cluster"])
```

In [274]:
```
print(b.first)
```

```
<bound method GroupBy.first of <pandas.core.groupby.generic.DataFrameGroupBy ob
ject at 0x0000026297F61670>>
```

In [275]:
```
a.value_counts()
```

Out[275]:
```
cluster
3          250
2          250
1          250
0          250
dtype: int64
```

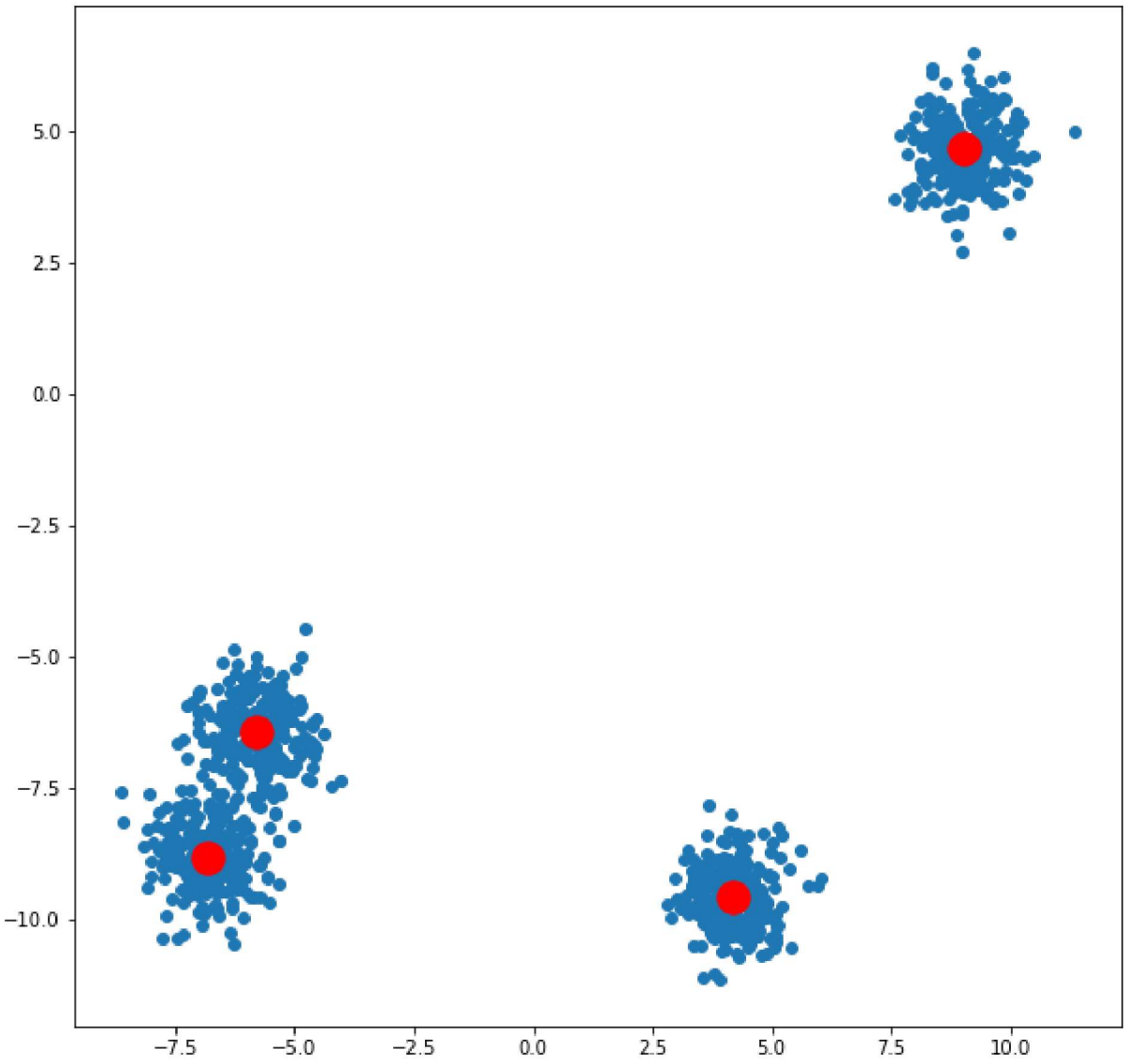
In [276]:
```
# just for practice
a["new"]=0
print(a)
a.value_counts()
```

```
     cluster  new
0          0    0
1          0    0
2          1    0
3          1    0
4          0    0
..       ...  ...
995        2    0
996        3    0
997        3    0
998        2    0
999        1    0

[1000 rows x 2 columns]
```

Out[276]:
```
cluster  new
3        0      250
2        0      250
1        0      250
0        0      250
dtype: int64
```

In [277]:
```
plt.figure(figsize=(10,10))
plt.scatter(X[:,1],X[:,2])
plt.scatter(kmeans.cluster_centers_[:,1],kmeans.cluster_centers_[:,2],c='red',s=
```

Out[277]: <matplotlib.collections.PathCollection at 0x26297faaa90>



In [ ]: