In [1]:

```python
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sn
import sklearn.cluster as cluster
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn import metrics
```

In [2]:

```python
df= pd.read_csv ('/Users/SAURABH/Saurabh patil/DATA SCIENCE/Clustering/EastWestAirlines.csv')
```

In [3]:

```python
df.head()
```

Out[3]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_miles_12mo	Flight_trans_12	Days_since_enroll	Award?
0	1	28143	0	1	1	1	174	1	0	0	7000	0
1	2	19244	0	1	1	1	215	2	0	0	6968	0
2	3	41354	0	1	1	1	4123	4	0	0	7034	0
3	4	14776	0	1	1	1	500	1	0	0	6952	0
4	5	97752	0	4	1	1	43300	26	2077	4	6935	1

In [4]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID#                3999 non-null   int64
 1   Balance            3999 non-null   int64
 2   Qual_miles         3999 non-null   int64
 3   cc1_miles          3999 non-null   int64
 4   cc2_miles          3999 non-null   int64
 5   cc3_miles          3999 non-null   int64
 6   Bonus_miles        3999 non-null   int64
 7   Bonus_trans        3999 non-null   int64
 8   Flight_miles_12mo  3999 non-null   int64
 9   Flight_trans_12    3999 non-null   int64
 10  Days_since_enroll  3999 non-null   int64
 11  Award?             3999 non-null   int64
dtypes: int64(12)
memory usage: 375.0 KB
```

In [5]:

```python
df.drop(['ID#'],axis=1, inplace=True)
```

In [6]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Balance            3999 non-null   int64
 1   Qual_miles         3999 non-null   int64
 2   cc1_miles          3999 non-null   int64
 3   cc2_miles          3999 non-null   int64
 4   cc3_miles          3999 non-null   int64
 5   Bonus_miles        3999 non-null   int64
 6   Bonus_trans        3999 non-null   int64
 7   Flight_miles_12mo  3999 non-null   int64
 8   Flight_trans_12    3999 non-null   int64
 9   Days_since_enroll  3999 non-null   int64
 10  Award?             3999 non-null   int64
dtypes: int64(11)
memory usage: 343.7 KB
```

In [7]:

```python
# Normalization function
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return (x)
```

In [8]:

```python
# Normalized data frame (considering the numerical part of data)
df_norm = norm_func(df.iloc[:,:])
```

In [10]:

```python
X = df_norm
```

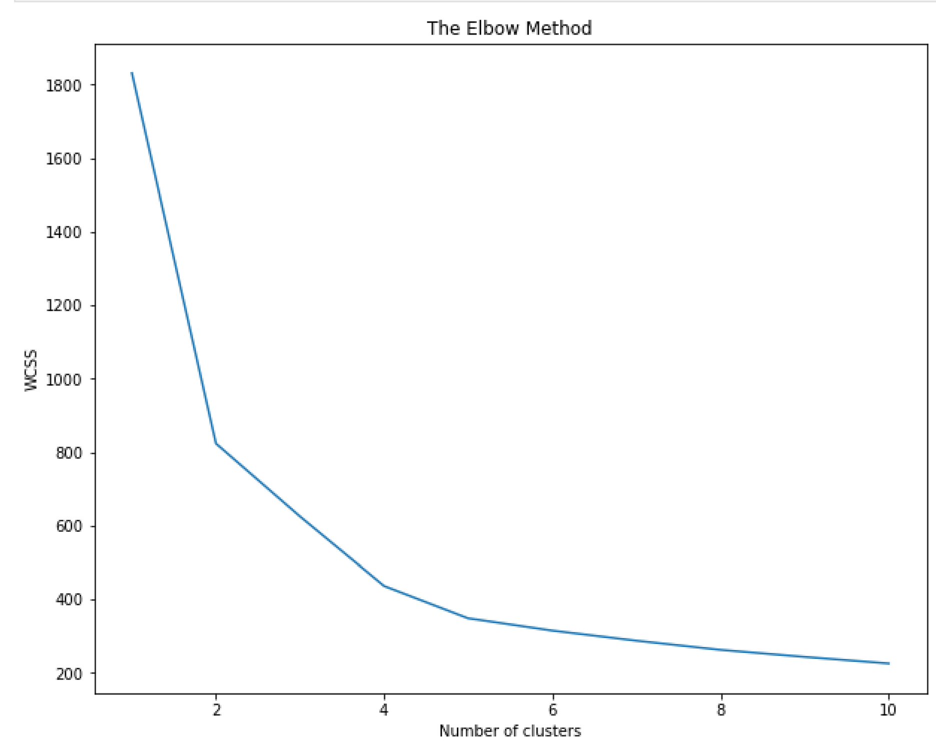
K Means Clustering

Using with Elbow method to find the optimum no of clusters

In [11]:

```python
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 200)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 8))
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



As per this plot, optimum no of clusters = 4

To confirm the same, let's use Silhouette score method

In [12]:

```python
for i in range(3,13):
    labels=cluster.KMeans(n_clusters=i,init="k-means++",random_state=200).fit(X).labels_
    print ("Silhouette score for k(clusters) = "+str(i)+" is "
           +str(metrics.silhouette_score(X,labels,metric="euclidean",sample_size=1000,random_state=200)))
```

```
Silhouette score for k(clusters) = 3 is 0.45748332355557725
Silhouette score for k(clusters) = 4 is 0.46521656243534165
Silhouette score for k(clusters) = 5 is 0.4331034289974864
Silhouette score for k(clusters) = 6 is 0.38813667045469546
Silhouette score for k(clusters) = 7 is 0.385463276140103
Silhouette score for k(clusters) = 8 is 0.37971466298124307
Silhouette score for k(clusters) = 9 is 0.3831682748940209
Silhouette score for k(clusters) = 10 is 0.3390167121162897
Silhouette score for k(clusters) = 11 is 0.3378168569656778
Silhouette score for k(clusters) = 12 is 0.34893258639417274
```

As per the Silhouette score also, K= 4 is the optimum number of clusters

In [13]:

```python
#Hence
model=KMeans(n_clusters=4)
model.fit(df_norm)
model.labels_
```

Out[13]: `array([1, 1, 1, ..., 3, 1, 1])`

In [14]:

```python
km = pd.Series(model.labels_)
df['kclust']= km
df.iloc[:,1:7].groupby(df.kclust).mean()
```

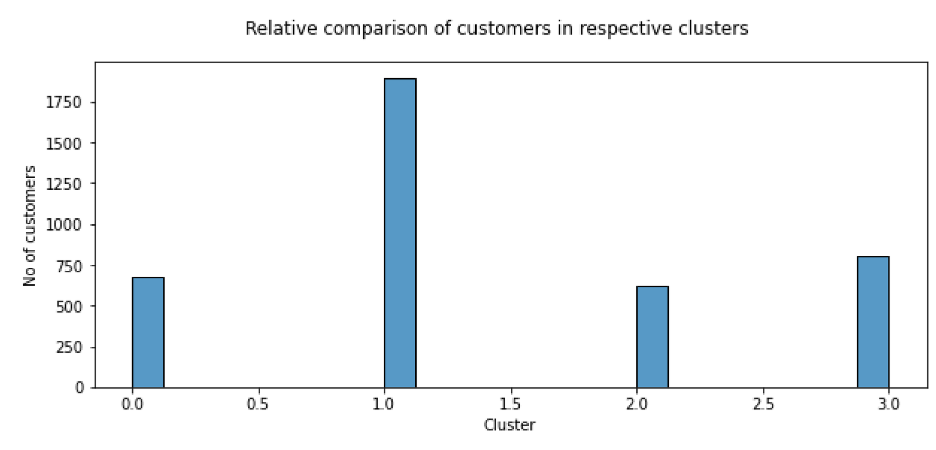
Out[14]:

kclust	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
0	290.453195	1.156018	1.032689	1.008915	8850.395245	10.476969
1	92.345809	1.092778	1.017923	1.004217	3335.152873	6.488666
2	73.112721	3.576490	1.001610	1.022544	31282.666667	17.251208
3	198.336634	3.915842	1.001238	1.025990	45609.657178	20.201733

In [15]:

```python
plt.figure(figsize=(10,4))
sns.histplot (x='kclust', data=df)
plt.xlabel('Cluster')
plt.ylabel('No of customers')
plt.suptitle('Relative comparison of customers in respective clusters')
```

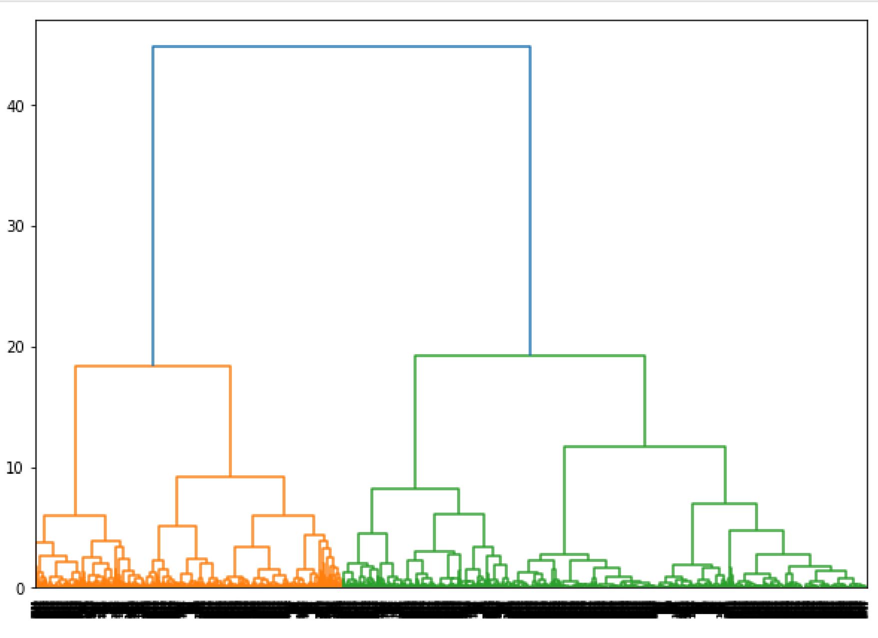
Out[15]: `Text(0.5, 0.98, 'Relative comparison of customers in respective clusters')`



Hierarchical Clustering

In [16]:

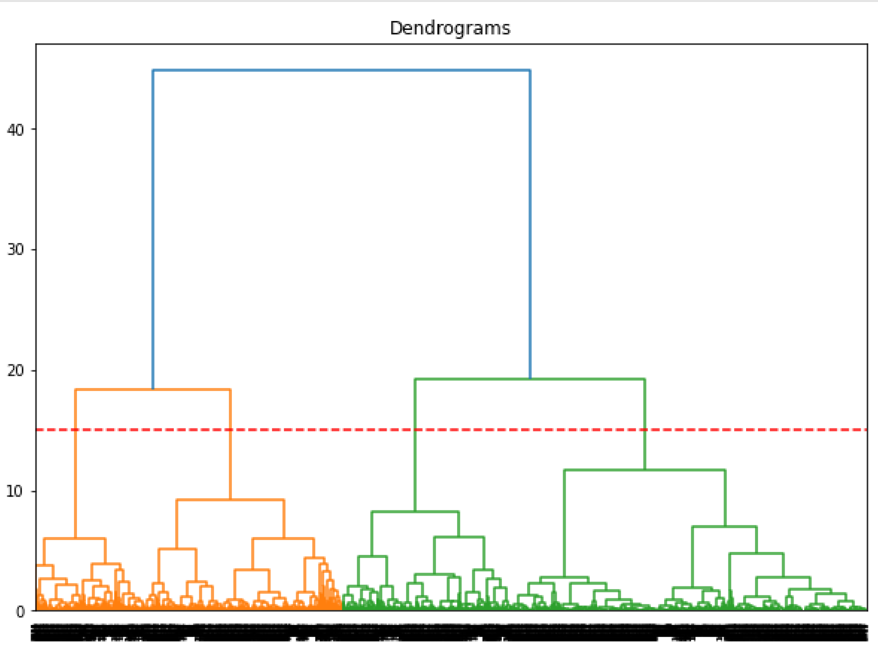
```python
# create dendrogram
plt.figure(figsize=(10, 7))
dendrogram = sch.dendrogram(sch.linkage(X, method='ward'))
```



The x-axis contains the samples and y-axis represents the distance between these samples. The vertical line with maximum distance is the blue line. If we decide a threshold of 15 and cut the dendrogram:

In [17]:

```python
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = sch.dendrogram(sch.linkage(X, method='ward'))
plt.axhline(y=15, color='r', linestyle='--')
plt.show()
```



In [18]:

```python
# create clusters
hc = AgglomerativeClustering(n_clusters=4, affinity = 'euclidean', linkage = 'single')
```

In [19]:

```python
y_hc = hc.fit_predict(df)
Clusters=pd.DataFrame(y_hc,columns=['Clusters'])
```

In [21]:

```python
Clusters.value_counts()
```

Out[21]:

```
Clusters
0           3994
1              3
3              1
2              1
dtype: int64
```

In [22]:

```python
Clusters
```

Out[22]:

	Clusters
0	0
1	0
2	0
3	0
4	0
...	...
3994	0
3995	0
3996	0
3997	0
3998	0

3999 rows × 1 columns

In [23]:

```python
df['hc_clust']= Clusters
df.iloc[:,1:7].groupby(df.hc_clust).mean()
```

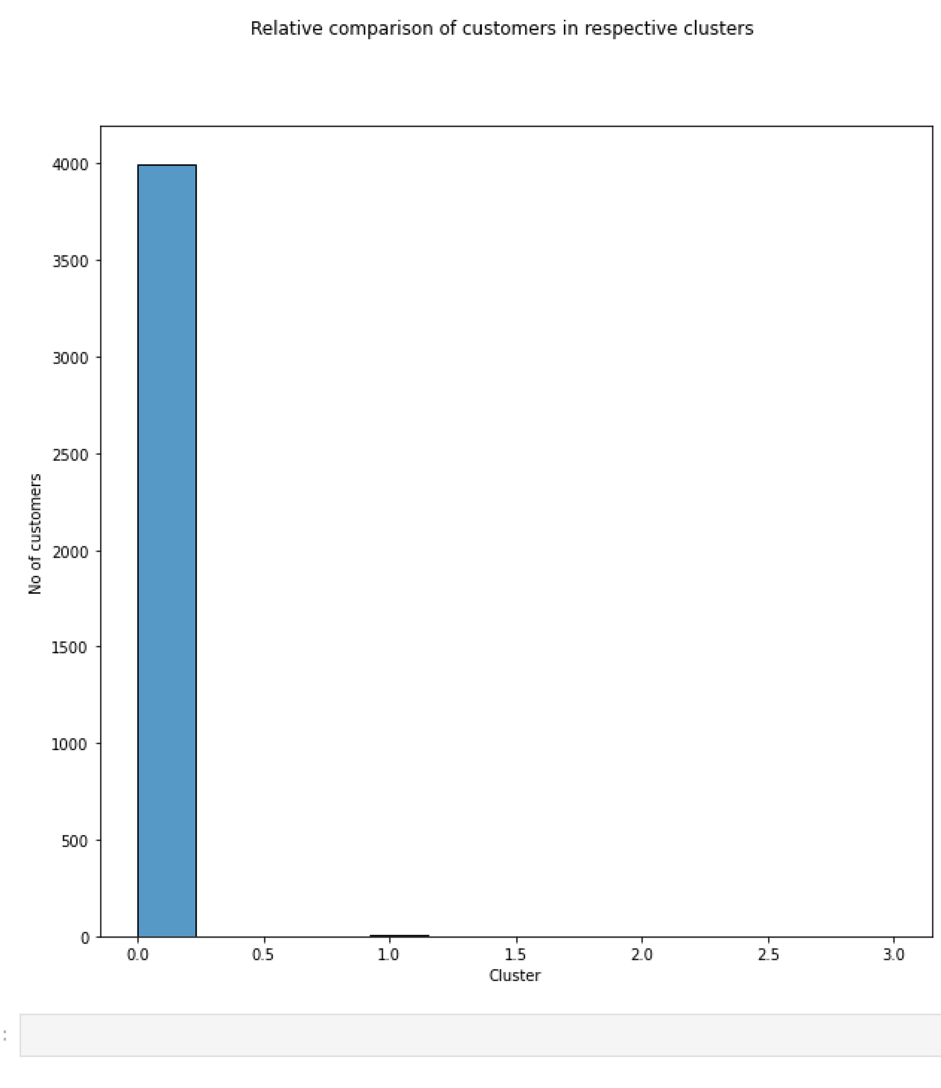
Out[23]:

hc_clust	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
0	142.382073	2.058338	1.014522	1.012268	17117.363545	11.585628
1	1644.666667	3.000000	1.000000	1.000000	29243.000000	19.666667
2	2706.000000	5.000000	1.000000	1.000000	90653.000000	32.000000
3	0.000000	1.000000	1.000000	1.000000	17108.000000	32.000000

In [24]:

```python
plt.figure(figsize=(10,10))
sns.histplot (x='hc_clust', data=df)
plt.xlabel('Cluster')
plt.ylabel('No of customers')
plt.suptitle('Relative comparison of customers in respective clusters')
```

Out[24]: `Text(0.5, 0.98, 'Relative comparison of customers in respective clusters')`



In [ ]: