Importing Necessary Liabraries

In [20]:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
import seaborn as sns
from sklearn.cluster import DBSCAN
```

Importing Dataset

In [13]:

```
airlines_data=pd.read_excel("EastWestAirlines.xlsx",sep=' ')
airlines_data
```

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_miles_
0	1	28143	0	1	1	1	174	1	
1	2	19244	0	1	1	1	215	2	
2	3	41354	0	1	1	1	4123	4	
3	4	14776	0	1	1	1	500	1	
4	5	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	...	
3994	4017	18476	0	1	1	1	8525	4	
3995	4018	64385	0	1	1	1	981	5	
3996	4019	73597	0	3	1	1	25447	8	
3997	4020	54899	0	1	1	1	500	1	
3998	4021	3016	0	1	1	1	0	0	

Initial analysis

In [14]:

```
airlines_data.shape
```

Out[14]:

```
(3999, 12)
```

In [15]:

```
airlines_data.isna().sum()
```

Out[15]:

```
ID#                  0
Balance              0
Qual_miles           0
cc1_miles            0
cc2_miles            0
cc3_miles            0
Bonus_miles          0
Bonus_trans          0
Flight_miles_12mo    0
Flight_trans_12      0
Days_since_enroll    0
Award?               0
dtype: int64
```

In [16]:

```
airlines_data.dtypes
```

Out[16]:

```
ID#                  int64
Balance              int64
Qual_miles           int64
cc1_miles            int64
cc2_miles            int64
cc3_miles            int64
Bonus_miles          int64
Bonus_trans          int64
Flight_miles_12mo    int64
Flight_trans_12      int64
Days_since_enroll    int64
Award?               int64
dtype: object
```

In [17]:

```
airlines1=airlines_data.drop(['ID#'],axis=1)
```

In [18]:

```
airlines1.head(10)
```

Out[18]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_mi
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
5	16420	0	1	1	1	0	0	
6	84914	0	3	1	1	27482	25	
7	20856	0	1	1	1	5250	4	
8	443003	0	3	2	1	1753	43	
9	104860	0	3	1	1	28426	28	

In [22]:

```
# Normalization function
airlines1_norm=pd.DataFrame(normalize(airlines1),columns=airlines1.columns)
airlines1_norm
```

Out[22]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Fligh
0	0.970414	0.0	0.000034	0.000034	0.000034	0.006000	0.000034	
1	0.940209	0.0	0.000049	0.000049	0.000049	0.010504	0.000098	
2	0.981113	0.0	0.000024	0.000024	0.000024	0.097817	0.000095	
3	0.904428	0.0	0.000061	0.000061	0.000061	0.030605	0.000061	
4	0.912226	0.0	0.000037	0.000009	0.000009	0.404078	0.000243	
...	...	...	...	...	...	...	...	
3994	0.905810	0.0	0.000049	0.000049	0.000049	0.417949	0.000196	
3995	0.999649	0.0	0.000016	0.000016	0.000016	0.015231	0.000078	
3996	0.944948	0.0	0.000039	0.000013	0.000013	0.326726	0.000103	
3997	0.999592	0.0	0.000018	0.000018	0.000018	0.009104	0.000018	
3998	0.907271	0.0	0.000301	0.000301	0.000301	0.000000	0.000000	

3999 rows × 11 columns

In [8]:

```
# How to find optimum number of  cluster
#The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluste
```

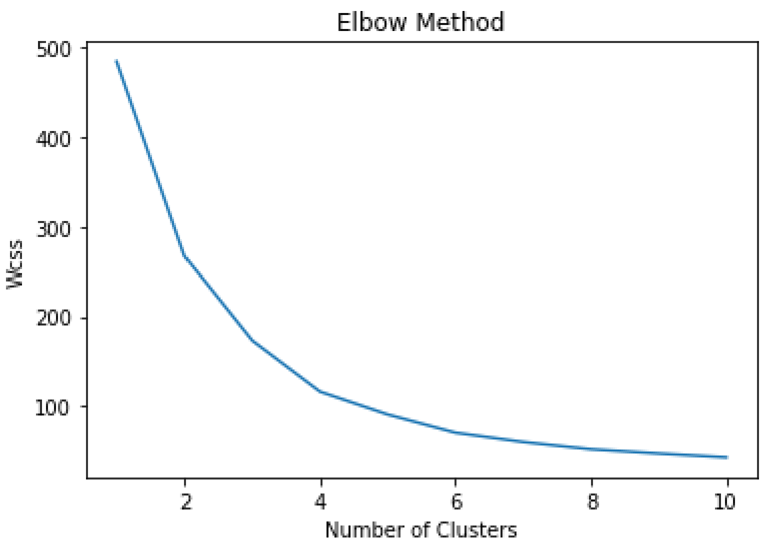
KMeans Clustering

In [23]:

```
wcss=[]
for i in range(1,11):
    kmeans=KMeans( n_clusters=i,random_state=0)
    kmeans.fit(airlines1_norm)
    wcss.append(kmeans.inertia_)
    print(wcss)

plt.plot(range(1,11),wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

```
[484.8511291307252]
[484.8511291307252, 267.597266143907]
[484.8511291307252, 267.597266143907, 173.27025625511413]
[484.8511291307252, 267.597266143907, 173.27025625511413, 116.3248160068040
6]
[484.8511291307252, 267.597266143907, 173.27025625511413, 116.3248160068040
6, 90.8239863037296]
[484.8511291307252, 267.597266143907, 173.27025625511413, 116.3248160068040
6, 90.8239863037296, 70.47242586480178]
[484.8511291307252, 267.597266143907, 173.27025625511413, 116.3248160068040
6, 90.8239863037296, 70.47242586480178, 60.072738651170795]
[484.8511291307252, 267.597266143907, 173.27025625511413, 116.3248160068040
6, 90.8239863037296, 70.47242586480178, 60.072738651170795, 51.9351665081741
2]
[484.8511291307252, 267.597266143907, 173.27025625511413, 116.3248160068040
6, 90.8239863037296, 70.47242586480178, 60.072738651170795, 51.9351665081741
2, 47.17112710270718]
[484.8511291307252, 267.597266143907, 173.27025625511413, 116.3248160068040
6, 90.8239863037296, 70.47242586480178, 60.072738651170795, 51.9351665081741
2, 47.17112710270718, 42.8711953420418]
```



Building K=3

In [38]:

```
#Build Cluster algorithm
# Cluster algorithm using K=3
clusters3=KMeans(3,random_state=12).fit(airlines1_norm)
clusters3
```

Out[38]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=12, tol=0.0001, verbose=0)
```

In [39]:

```
clusters3.labels_
```

Out[39]:

```
array([1, 1, 1, ..., 1, 1, 1])
```

In [40]:

```
# Assign clusters to the data set
airlines3=airlines1.copy()
airlines3['clusters3']=clusters3.labels_
airlines3
```

Out[40]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
3994	18476	0	1	1	1	8525	4	
3995	64385	0	1	1	1	981	5	
3996	73597	0	3	1	1	25447	8	
3997	54899	0	1	1	1	500	1	
3998	3016	0	1	1	1	0	0	

3999 rows × 12 columns

In [41]:

```
clusters3.cluster_centers_
```

Out[41]:

```
array([[6.30508213e-01, 9.15231468e-04, 2.08191153e-04, 2.07409255e-04,
        2.07409255e-04, 1.22266715e-01, 4.68663039e-04, 6.69160773e-03,
        2.24693038e-05, 6.85884357e-01, 2.55697749e-05],
       [9.72915814e-01, 3.32753701e-03, 4.32515674e-05, 3.56632121e-05,
        3.54159133e-05, 1.33941084e-01, 2.11254052e-04, 6.76322602e-03,
        2.11717605e-05, 9.85574060e-02, 5.46966912e-06],
       [7.02585534e-01, 2.28699642e-03, 7.65752681e-05, 3.73696106e-05,
        3.65177677e-05, 6.40527156e-01, 4.48000121e-04, 1.23850237e-02,
        4.00021547e-05, 1.06746997e-01, 2.03293904e-05]])
```

In [42]:

```
# Group data by Clusters (K=3)
airlines3.groupby('clusters3').agg(['mean']).reset_index()
```

Out[42]:

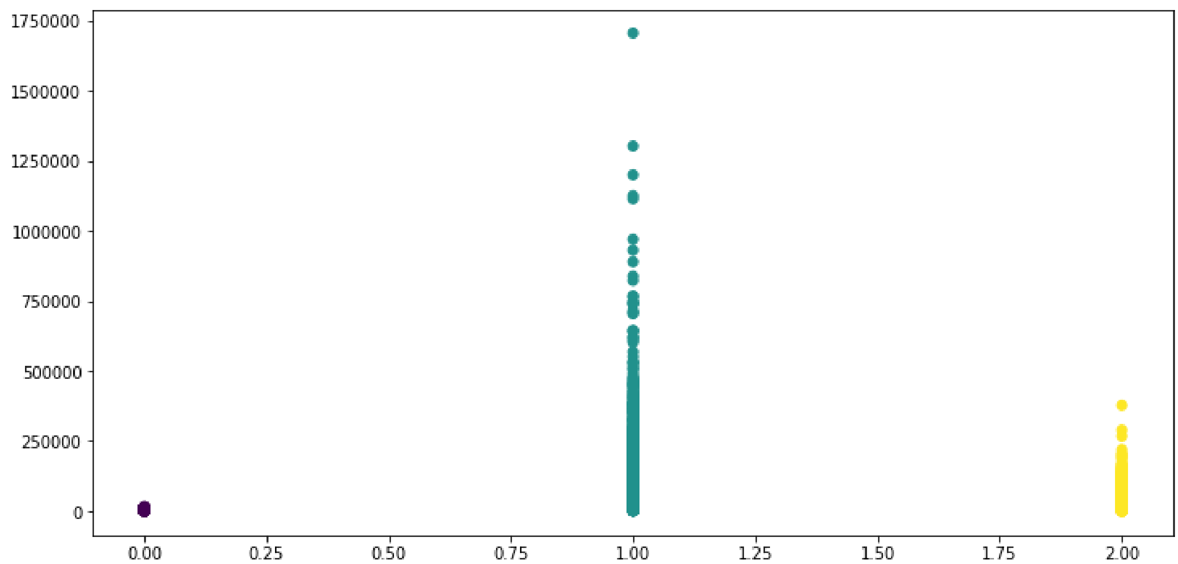
	clusters3	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_
		mean	mean	mean	mean	mean	mean	mean
0	0	5131.617886	8.150407	1.008130	1.000000	1.000000	869.394309	2.9:
1	1	86696.132743	164.510551	1.798162	1.011232	1.001702	12337.173928	10.5!
2	2	47062.690798	111.628221	3.319018	1.030675	1.053988	39388.652761	17.8!

In [43]:

```
# Plot Clusters
plt.figure(figsize=(12, 6))
plt.scatter(airlines3['clusters3'],airlines3['Balance'], c=clusters3.labels_)
```

Out[43]:

<matplotlib.collections.PathCollection at 0x20eaee455c8>



Build clusters K=5

In [32]:

```
# Cluster algorithm using K=5
clusters5=KMeans(5,random_state=12).fit(airlines1_norm)
clusters5
```

Out[32]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=12, tol=0.0001, verbose=0)
```

In [33]:

```
clusters5.labels_
```

Out[33]:

```
array([0, 2, 0, ..., 4, 0, 2])
```

In [34]:

```
# Assign clusters to the data set
airlines5=airlines1.copy()
airlines5['clusters5']=clusters5.labels_
airlines5
```

Out[34]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_mile
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
94	18476	0	1	1	1	8525	4	
95	64385	0	1	1	1	981	5	
96	73597	0	3	1	1	25447	8	
97	54899	0	1	1	1	500	1	
98	3016	0	1	1	1	0	0	

9 rows × 12 columns

In [35]:

```
# Compute the centroids for K=5 clusters with 11 variables
clusters5.cluster_centers_
```

Out[35]:

```
array([[9.87347192e-01, 3.41837203e-03, 3.52620775e-05, 3.03479065e-05,
        3.02349363e-05, 9.15517711e-02, 1.54818686e-04, 6.61324432e-03,
        2.08325733e-05, 7.54676163e-02, 3.99111816e-06],
       [5.14097044e-01, 2.46403313e-03, 9.56772813e-05, 5.01782621e-05,
        4.88674224e-05, 8.02764990e-01, 5.20805294e-04, 1.79689628e-02,
        6.06455235e-05, 1.36723853e-01, 3.06681430e-05],
       [8.92936852e-01, 4.46454511e-03, 1.23968035e-04, 1.23783403e-04,
        1.23783403e-04, 7.58365867e-02, 2.93996886e-04, 6.32105922e-03,
        2.08016784e-05, 4.07924096e-01, 1.35510886e-05],
       [4.14644791e-01, 1.30104261e-18, 2.28611980e-04, 2.27627266e-04,
        2.27627266e-04, 1.50766683e-01, 5.97513433e-04, 7.35401490e-03,
        2.84888383e-05, 8.48268382e-01, 3.91049405e-05],
       [8.90527678e-01, 1.91114047e-03, 5.81321812e-05, 3.02250715e-05,
        2.95023684e-05, 4.23591789e-01, 4.07332101e-04, 7.83191039e-03,
        2.30605390e-05, 8.30790414e-02, 1.00466076e-05]])
```

In [36]:

```
# Group data by Clusters (K=5)
airlines5.groupby('clusters5').agg(['mean']).reset_index()
```

Out[36]:

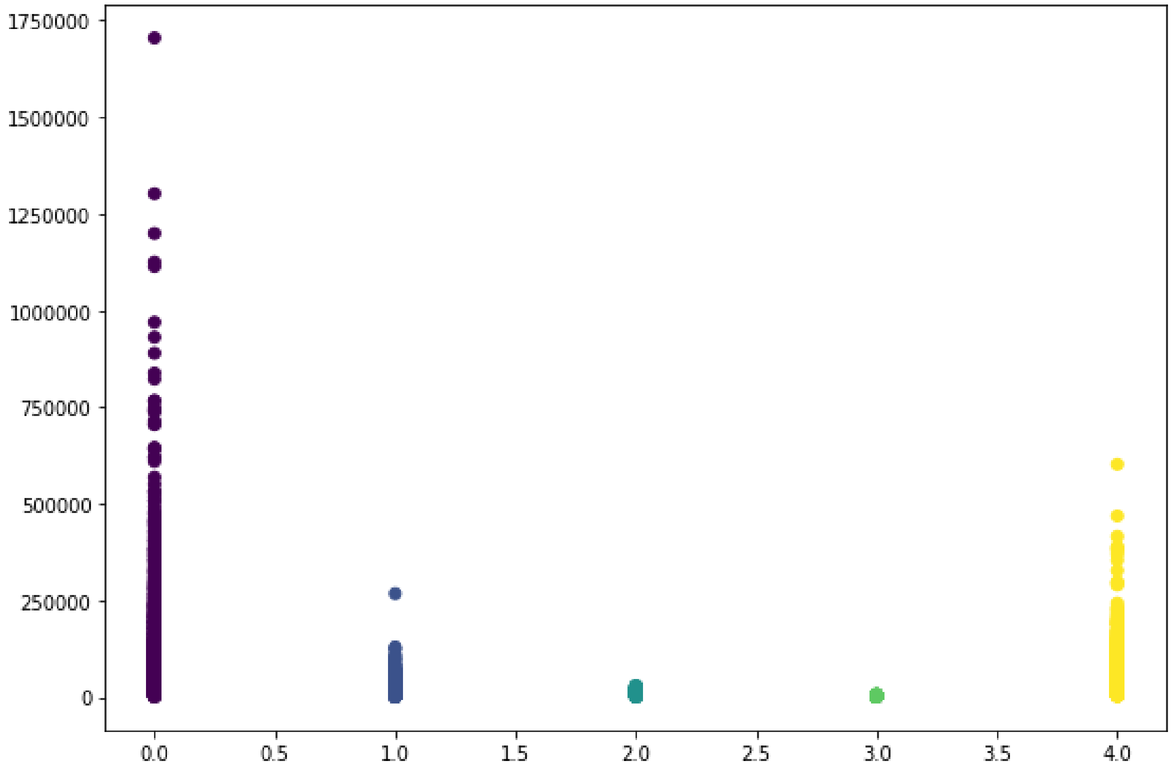
	clusters5	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_
		mean	mean	mean	mean	mean	mean	mean
0	0	97077.624478	187.605934	1.610570	1.009272	1.001854	9664.546129	9.7
1	1	27462.797721	116.148148	3.245014	1.034188	1.071225	41806.162393	17.5
2	2	11756.307494	55.263566	1.005168	1.000000	1.000000	980.863049	3.4
3	3	2415.576577	0.000000	1.009009	1.000000	1.000000	850.189189	3.0
4	4	70974.834844	110.264854	3.144008	1.026183	1.020141	32797.468278	17.7

In [37]:

```
# Plot Clusters
plt.figure(figsize=(10, 7))
plt.scatter(airlines5['clusters5'],airlines5['Balance'], c=clusters5.labels_)
```

Out[37]:

```
<matplotlib.collections.PathCollection at 0x20eaebacf48>
```



Hierarchial Clustering

In [47]:

```
airlines_data=pd.read_excel('EastWestAirlines.xlsx')
airlines_data
```

Out[47]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_tra
0	1	28143	0	1	1	1	174	
1	2	19244	0	1	1	1	215	
2	3	41354	0	1	1	1	4123	
3	4	14776	0	1	1	1	500	
4	5	97752	0	4	1	1	43300	
...	...	...	...	...	...	...	...	
3994	4017	18476	0	1	1	1	8525	
3995	4018	64385	0	1	1	1	981	
3996	4019	73597	0	3	1	1	25447	
3997	4020	54899	0	1	1	1	500	
3998	4021	3016	0	1	1	1	0	

3999 rows × 12 columns

In [48]:

```
airline1=airlines1
```

In [49]:

```
airline1.head()
```

Out[49]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_mi
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	

In [50]:

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

In [51]:

```
# Normalized data frame (considering the numerical part of data)
airline1_norm = norm_func(airline1)
airline1_norm
```

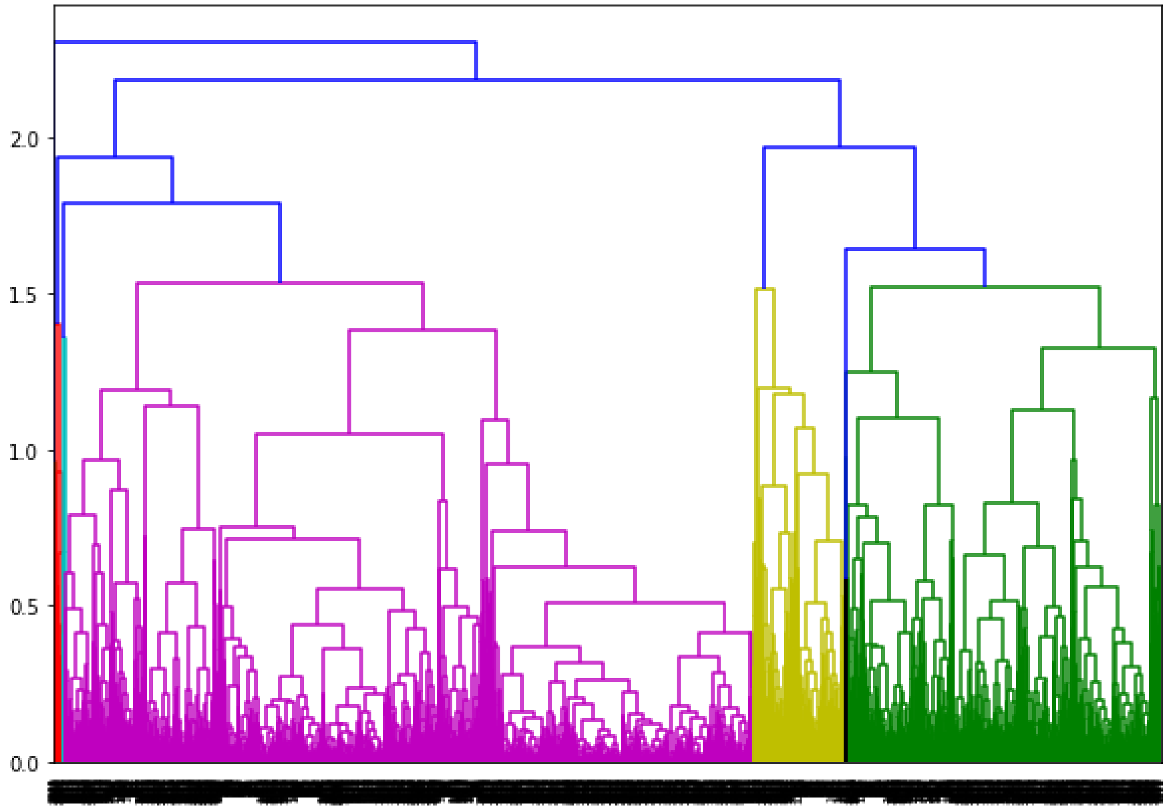
Out[51]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Fligh
0	0.016508	0.0	0.00	0.0	0.0	0.000660	0.011628	
1	0.011288	0.0	0.00	0.0	0.0	0.000815	0.023256	
2	0.024257	0.0	0.00	0.0	0.0	0.015636	0.046512	
3	0.008667	0.0	0.00	0.0	0.0	0.001896	0.011628	
4	0.057338	0.0	0.75	0.0	0.0	0.164211	0.302326	
...	...	...	...	...	...	...	...	
3994	0.010837	0.0	0.00	0.0	0.0	0.032330	0.046512	
3995	0.037766	0.0	0.00	0.0	0.0	0.003720	0.058140	
3996	0.043169	0.0	0.50	0.0	0.0	0.096505	0.093023	
3997	0.032202	0.0	0.00	0.0	0.0	0.001896	0.011628	
3998	0.001769	0.0	0.00	0.0	0.0	0.000000	0.000000	

3999 rows × 11 columns

In [55]:

```
# Create Dendrograms
plt.figure(figsize=(10, 7))
dendograms=sch.dendrogram(sch.linkage(airline1_norm,'complete'))
```



In [56]:

```
# Create Clusters (y)
hclusters=AgglomerativeClustering(n_clusters=5,affinity='euclidean',linkage='ward')
hclusters
```

Out[56]:

```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                        connectivity=None, distance_threshold=None,
                        linkage='ward', memory=None, n_clusters=5,
                        pooling_func='deprecated')
```

In [57]:

```
y=pd.DataFrame(hclusters.fit_predict(airline1_norm),columns=['clusters'])
y['clusters'].value_counts()
```

Out[57]:

```
1    1011
0     946
2     808
4     699
3     535
Name: clusters, dtype: int64
```

In [58]:

```
# Adding clusters to dataset
airline1['clusters']=hclusters.labels_
airline1
```

Out[58]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_mi
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
994	18476	0	1	1	1	8525	4	
995	64385	0	1	1	1	981	5	
996	73597	0	3	1	1	25447	8	
997	54899	0	1	1	1	500	1	
998	3016	0	1	1	1	0	0	

)99 rows × 12 columns

In [59]:

```
airline1.groupby('clusters').agg(['mean']).reset_index()
```

Out[59]:

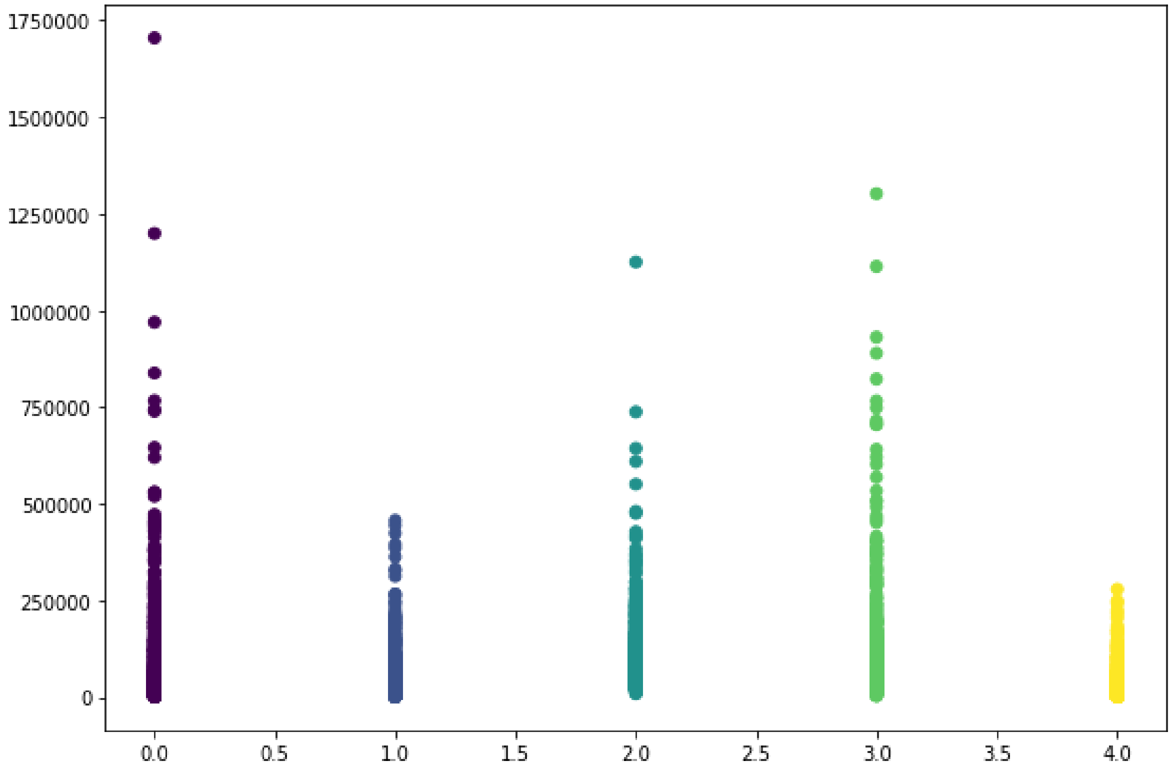
	clusters	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_
		mean	mean	mean	mean	mean	mean	mean
0	0	79848.233615	285.097252	1.699789	1.024313	1.000000	12079.774841	12.1
1	1	43313.653808	21.506429	1.000000	1.033630	1.000989	2562.614243	5.4
2	2	106221.111386	161.262376	3.198020	1.001238	1.025990	26458.257426	16.3
3	3	127475.028037	160.801869	4.362617	1.000000	1.050467	58656.919626	22.2
4	4	30013.416309	98.054363	1.000000	1.000000	1.000000	2552.569385	6.1

In [60]:

```
#Plot Clusters
plt.figure(figsize=(10, 7))
plt.scatter(airline1['clusters'],airline1['Balance'], c=hclusters.labels_)
```

Out[60]:

```
<matplotlib.collections.PathCollection at 0x20eae6d59c8>
```



DBSCAN Clustering

In [64]:

```
airlines1.head()
```

Out[64]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_mi
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	

In [63]:

```
del airlines1['clusters']
```

In [65]:

```
# Normalize heterogenous numerical data using standard scalar fit transform to dataset
airlines_norm=StandardScaler().fit_transform(airlines1)
airlines_norm
```

Out[65]:

```
array([[-4.51140783e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.39545434e+00, -7.66919299e-01],
       [-5.39456874e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.37995704e+00, -7.66919299e-01],
       [-3.20031232e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.41192021e+00, -7.66919299e-01],
       ...,
       [-4.29480975e-05, -1.86298687e-01,  6.83121167e-01, ...,
        -3.62167870e-01, -1.31560393e+00,  1.30391816e+00],
       [-1.85606976e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -9.85033311e-02, -1.31608822e+00, -7.66919299e-01],
       [-7.00507951e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01, -1.31754109e+00, -7.66919299e-01]])
```

In [71]:

```
# DBSCAN Clustering
dbscan=DBSCAN(eps=1,min_samples=3)
dbscan.fit(airlines_norm)
```

Out[71]:

```
DBSCAN(algorithm='auto', eps=1, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=3, n_jobs=None, p=None)
```

In [72]:

```
dbscan.labels_
```

Out[72]:

```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

In [73]:

```
# Adding clusters to dataset
airlines1['clusters']=dbscan.labels_
airlines1
```

Out[73]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_mi
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	
...	...	...	...	...	...	...	...	
994	18476	0	1	1	1	8525	4	
995	64385	0	1	1	1	981	5	
996	73597	0	3	1	1	25447	8	
997	54899	0	1	1	1	500	1	
998	3016	0	1	1	1	0	0	

)99 rows × 12 columns

In [74]:

```
airlines1.groupby('clusters').agg(['mean']).reset_index()
```

Out[74]:

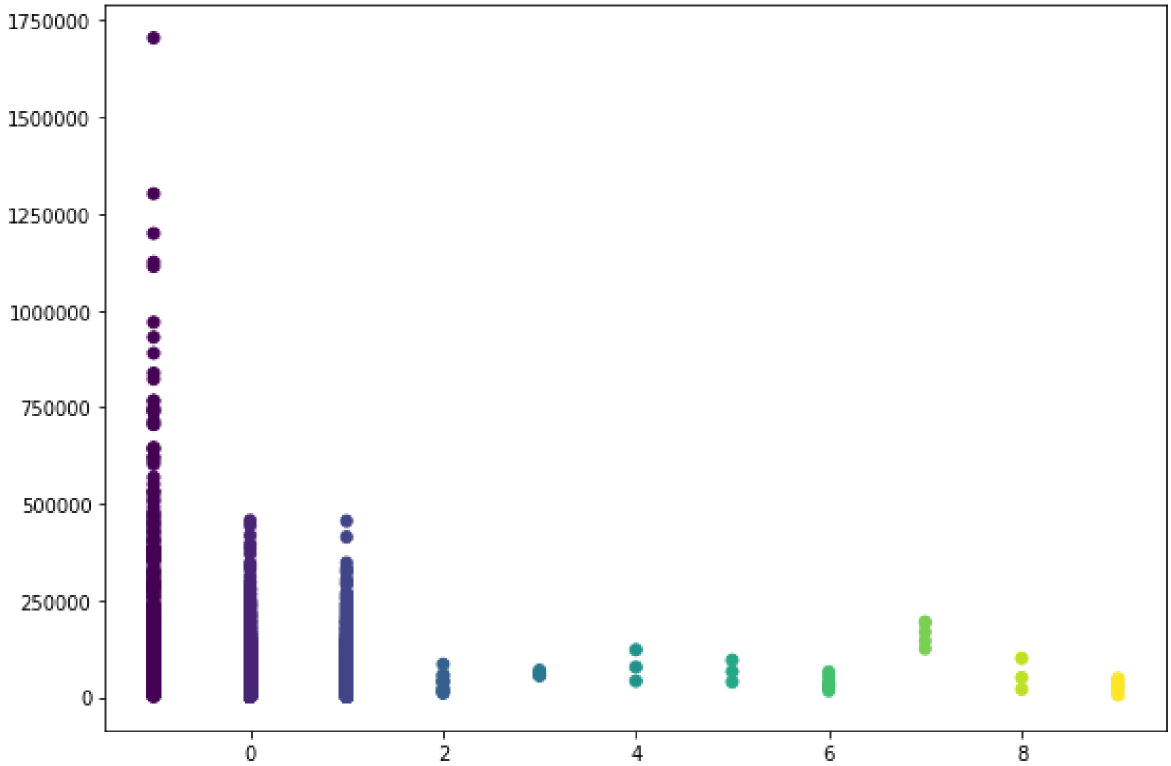
	clusters	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonu
		mean	mean	mean	mean	mean	mean	mear
0	-1	186537.458661	999.370079	2.777559	1.061024	1.096457	41827.370079	22
1	0	54311.522669	5.073139	1.663388	1.000000	1.000000	9127.850727	8
2	1	63282.187330	9.723982	2.589140	1.000000	1.000000	22949.240724	12
3	2	34806.538462	0.000000	1.000000	2.000000	1.000000	8389.769231	12
4	3	60932.000000	1794.500000	3.750000	1.000000	1.000000	39889.750000	16
5	4	80078.000000	0.000000	3.000000	1.000000	1.000000	15252.333333	19
6	5	66728.666667	0.000000	2.666667	1.000000	1.000000	30841.333333	28
7	6	36413.428571	0.000000	1.000000	3.000000	1.000000	14341.142857	13
8	7	164883.400000	1471.600000	1.000000	1.000000	1.000000	8472.800000	5
9	8	56459.000000	2486.333333	2.333333	1.000000	1.000000	12619.000000	11
10	9	27820.200000	2403.300000	1.100000	1.000000	1.000000	2659.400000	5

In [75]:

```
# Plot Clusters
plt.figure(figsize=(10, 7))
plt.scatter(airlines1['clusters'],airlines1['Balance'], c=dbscan.labels_)
```

Out[75]:

```
<matplotlib.collections.PathCollection at 0x20eb3c6b7c8>
```



>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>The End!! <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<