In [1]:

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from numpy import random,float,array
import numpy as np
import seaborn as sns
```

In [3]:

```
ewa=pd.read_csv('EastWestAirlines.csv')
```

In [4]:

```
ewa.head()
```

Out[4]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flig
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	

In [5]:

```
ewa.shape
```

Out[5]:

```
(3999, 12)
```

In [6]:

```
def norm_fun(i):
    x=(i-i.min())/(i.max()-i.min())
    return(x)
```

In [7]:

```
df_norm=norm_fun(ewa.iloc[:,1:])
df_norm.describe()
```

Out[7]:

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_
count	3999.000000	3999.000000	3999.000000	3999.000000	3999.000000	3999.000000	3999.00
mean	0.043172	0.012927	0.264879	0.007252	0.003063	0.065020	0.13
std	0.059112	0.069399	0.344230	0.073825	0.048810	0.091590	0.1
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.010868	0.000000	0.000000	0.000000	0.000000	0.004741	0.03
50%	0.025279	0.000000	0.000000	0.000000	0.000000	0.027195	0.13
75%	0.054201	0.000000	0.500000	0.000000	0.000000	0.090261	0.19
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

In [8]:

```
from scipy.cluster.hierarchy import linkage

import scipy.cluster.hierarchy as sch
```

In [9]:

```
z=linkage(df_norm,method="complete",metric="euclidean")
```

In [10]:

```
from sklearn.preprocessing import MinMaxScaler
trans = MinMaxScaler()
data = pd.DataFrame(trans.fit_transform(ewa.iloc[:,1:]))
data
```

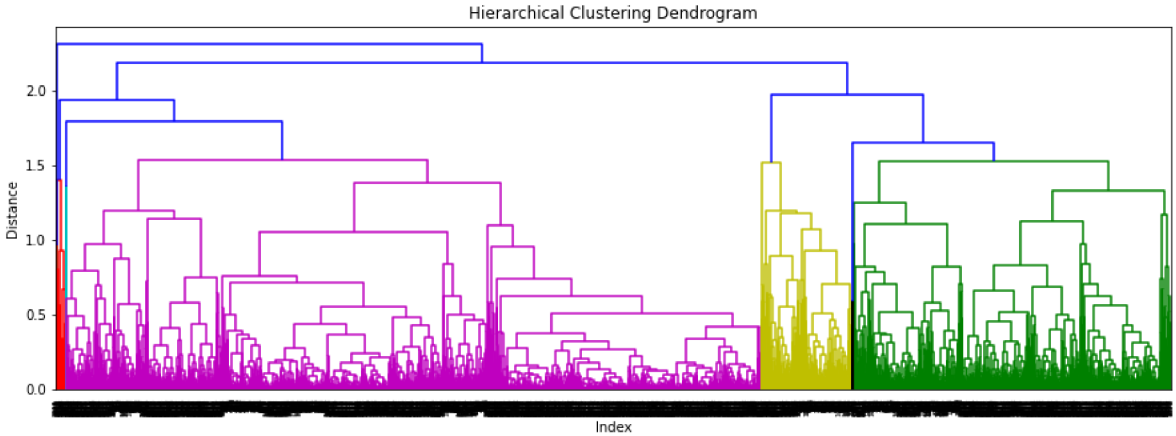
Out[10]:

	0	1	2	3	4	5	6	7	8	9	10
0	0.016508	0.0	0.00	0.0	0.0	0.000660	0.011628	0.000000	0.000000	0.843742	0.0
1	0.011288	0.0	0.00	0.0	0.0	0.000815	0.023256	0.000000	0.000000	0.839884	0.0
2	0.024257	0.0	0.00	0.0	0.0	0.015636	0.046512	0.000000	0.000000	0.847842	0.0
3	0.008667	0.0	0.00	0.0	0.0	0.001896	0.011628	0.000000	0.000000	0.837955	0.0
4	0.057338	0.0	0.75	0.0	0.0	0.164211	0.302326	0.067398	0.075472	0.835905	1.0
...	...	...	...	...	...	...	...	...	...	...	...
3994	0.010837	0.0	0.00	0.0	0.0	0.032330	0.046512	0.006490	0.018868	0.168917	1.0
3995	0.037766	0.0	0.00	0.0	0.0	0.003720	0.058140	0.000000	0.000000	0.167953	1.0
3996	0.043169	0.0	0.50	0.0	0.0	0.096505	0.093023	0.000000	0.000000	0.168797	1.0
3997	0.032202	0.0	0.00	0.0	0.0	0.001896	0.011628	0.016225	0.018868	0.168676	0.0
3998	0.001769	0.0	0.00	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.168314	0.0

3999 rows × 11 columns

In [11]:

```
from scipy.cluster.hierarchy import linkage
import scipy.cluster.hierarchy as sch # for creating dendrogram
#p = np.array(df_norm) # converting into numpy array format
z = linkage(df_norm, method="complete",metric="euclidean")
plt.figure(figsize=(15, 5))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Index')
plt.ylabel('Distance')
sch.dendrogram(
    z,
    #leaf_rotation=0.,  # rotates the x axis labels
    #leaf_font_size=8.,  # font size for the x axis labels
)
plt.show()
```



In [15]:

```
from sklearn.cluster import AgglomerativeClustering
h_complete = AgglomerativeClustering(n_clusters=5, linkage='complete',affinity = "euclidean

cluster_labels=pd.Series(h_complete.labels_)
cluster_labels
ewa['clust']=cluster_labels # creating a  new column and assigning it to new column
ewa
```

Out[15]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
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

3999 rows × 13 columns

In [16]:

```
ewa.iloc[:,1:].groupby(ewa.clust).mean()
```

Out[16]:

clust	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
0	59968.433667	88.883768	1.712224	1.000401	1.011222	10271.530261	9.105812
1	157084.578462	208.673846	4.661538	1.000000	1.061538	70477.086154	23.249231
2	80173.963287	248.550699	2.104895	1.009615	1.000874	16882.864510	13.412587
3	131999.500000	347.000000	2.500000	1.000000	1.000000	65634.250000	69.250000
4	45515.064516	32.258065	1.000000	2.483871	1.000000	14618.870968	16.129032

In [17]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
ewa_subset = pd.DataFrame(scaler.fit_transform(ewa.iloc[:,1:]))
ewa_subset
```

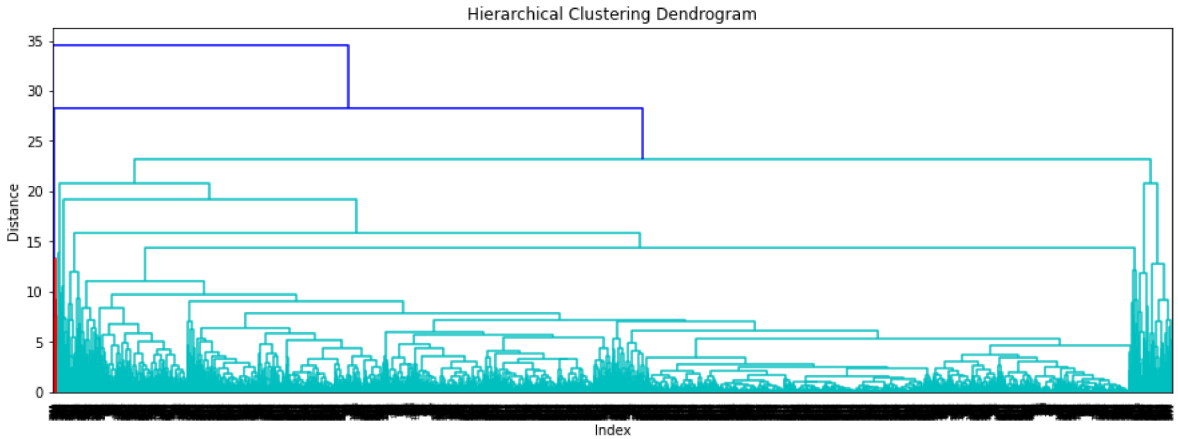
Out[17]:

	0	1	2	3	4	5	6	7	
0	-0.451141	-0.186299	-0.769578	-0.098242	-0.062767	-0.702786	-1.104065	-0.328603	-0.3
1	-0.539457	-0.186299	-0.769578	-0.098242	-0.062767	-0.701088	-0.999926	-0.328603	-0.3
2	-0.320031	-0.186299	-0.769578	-0.098242	-0.062767	-0.539253	-0.791649	-0.328603	-0.3
3	-0.583799	-0.186299	-0.769578	-0.098242	-0.062767	-0.689286	-1.104065	-0.328603	-0.3
4	0.239678	-0.186299	1.409471	-0.098242	-0.062767	1.083121	1.499394	1.154932	0.(
...	...	...	...	...	...	...	...	...	
3994	-0.547079	-0.186299	-0.769578	-0.098242	-0.062767	-0.356960	-0.791649	-0.185750	-0.(
3995	-0.091465	-0.186299	-0.769578	-0.098242	-0.062767	-0.669367	-0.687511	-0.328603	-0.3
3996	-0.000043	-0.186299	0.683121	-0.098242	-0.062767	0.343804	-0.375096	-0.328603	-0.3
3997	-0.185607	-0.186299	-0.769578	-0.098242	-0.062767	-0.689286	-1.104065	0.028531	-0.(
3998	-0.700508	-0.186299	-0.769578	-0.098242	-0.062767	-0.709992	-1.208203	-0.328603	-0.3

3999 rows × 12 columns

In [18]:

```
from scipy.cluster.hierarchy import linkage
import scipy.cluster.hierarchy as sch # for creating dendrogram
#p = np.array(df_norm) # converting into numpy array format
z = linkage(ewa_subset, method="complete",metric="euclidean")
plt.figure(figsize=(15, 5))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Index')
plt.ylabel('Distance')
sch.dendrogram(
    z,
    #leaf_rotation=0.,  # rotates the x axis labels
    #leaf_font_size=8.,  # font size for the x axis labels
)
plt.show()
```



In [21]:

```
from sklearn.cluster import AgglomerativeClustering
h_complete = AgglomerativeClustering(n_clusters=5, linkage='complete',affinity = "euclidean

cluster_labels=pd.Series(h_complete.labels_)
cluster_labels
ewa['clust']=cluster_labels # creating a  new column and assigning it to new column
ewa
```

Out[21]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
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

3999 rows × 13 columns

In [22]:

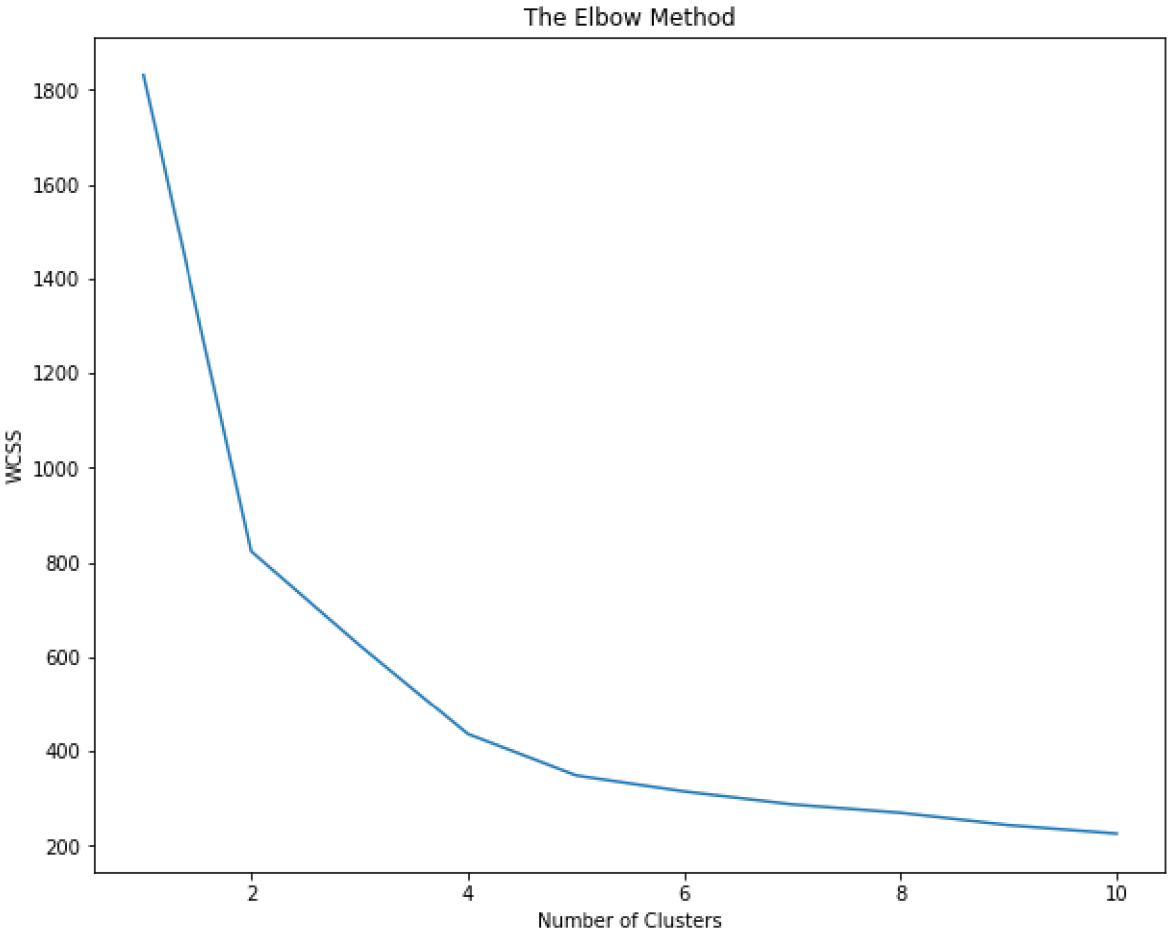
```
k=list(range(2,20))
```

In [23]:

```
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
import numpy as np
```

In [24]:

```
from sklearn.cluster import KMeans
fig = plt.figure(figsize=(10, 8))
WCSS = []
for i in range(1, 11):
    clf = KMeans(n_clusters=i)
    clf.fit(df_norm)
    WCSS.append(clf.inertia_) # inertia is another name for WCSS
plt.plot(range(1, 11), WCSS)
plt.title('The Elbow Method')
plt.ylabel('WCSS')
plt.xlabel('Number of Clusters')
plt.show()
```



In [25]:

```
clf = KMeans(n_clusters=5)
y_kmeans = clf.fit_predict(df_norm)
```

In [26]:

```
y_kmeans
#clf.cluster_centers_
clf.labels_
```

Out[26]:

```
array([4, 4, 4, ..., 3, 0, 0], dtype=int32)
```

In [28]:

```
md=pd.Series(y_kmeans)  # converting numpy array into pandas series object
ewa['clust']=md # creating a  new column and assigning it to new column
ewa.describe()
```

Out[28]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonu
count	3999.000000	3.999000e+03	3999.000000	3999.000000	3999.000000	3999.000000	399
mean	2014.819455	7.360133e+04	144.114529	2.059515	1.014504	1.012253	17144
std	1160.764358	1.007757e+05	773.663804	1.376919	0.147650	0.195241	24150
min	1.000000	0.000000e+00	0.000000	1.000000	1.000000	1.000000	0
25%	1010.500000	1.852750e+04	0.000000	1.000000	1.000000	1.000000	1250
50%	2016.000000	4.309700e+04	0.000000	1.000000	1.000000	1.000000	7171
75%	3020.500000	9.240400e+04	0.000000	3.000000	1.000000	1.000000	23800
max	4021.000000	1.704838e+06	11148.000000	5.000000	3.000000	5.000000	263685

In [29]:

```
ewa.iloc[:,1:7].groupby(ewa.clust).mean()
```

Out[29]:

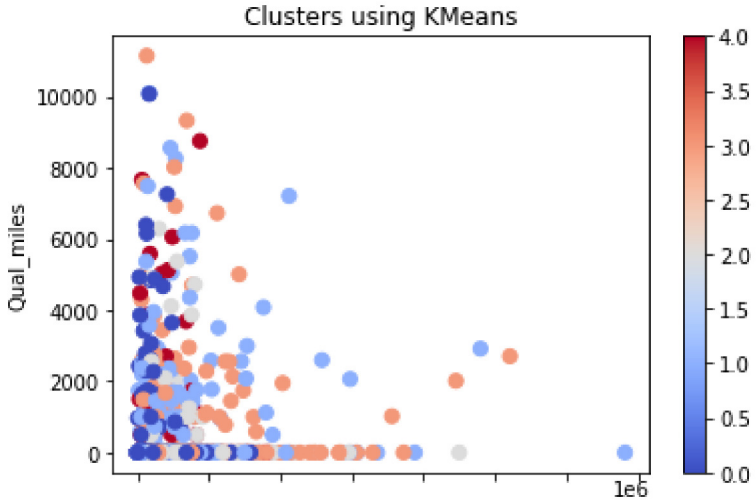
clust	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles
0	33097.301357	94.131783	1.070736	1.016473	1.006783	3244.520349
1	83529.153046	290.453195	1.156018	1.032689	1.008915	8850.395245
2	118297.325243	73.467638	3.584142	1.001618	1.022654	31384.393204
3	108317.387376	198.336634	3.915842	1.001238	1.025990	45609.657178
4	49921.633641	89.903226	1.122120	1.019585	1.001152	3467.074885

In [31]:

```
ewa.plot(x="Balance",y ="Qual_miles",c=clf.labels_,kind="scatter",s=50 ,cmap=plt.cm.coolwar
plt.title('Clusters using KMeans')
```

Out[31]:

```
Text(0.5, 1.0, 'Clusters using KMeans')
```

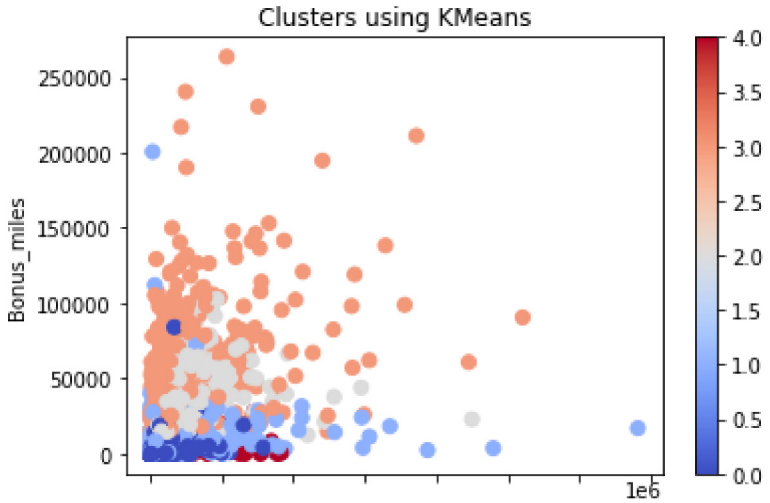


In [32]:

```
ewa.plot(x="Balance",y ="Bonus_miles",c=clf.labels_,kind="scatter",s=50 ,cmap=plt.cm.coolwa
plt.title('Clusters using KMeans')
```

Out[32]:

```
Text(0.5, 1.0, 'Clusters using KMeans')
```



In [33]:

```
clf.inertia_
```

Out[33]:

```
348.94342839013245
```

In [34]:

```
WCSS
```

Out[34]:

```
[1830.7932128584107,
 823.6756984125209,
 625.1693121408771,
 436.7088576193257,
 348.943321725414,
 315.3155964842893,
 287.73481183138455,
 270.09187065810477,
 243.92551718928684,
 226.11861635972758]
```

DBSCAN

In [35]:

```
#Import the libraries
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [37]:

```
# Import .csv file and convert it to a DataFrame object
df = pd.read_csv("EastWestAirlines.csv");

print(df.head())
df
```

```
   ID#  Balance  Qual_miles  ...  Flight_trans_12  Days_since_enroll  Award?
0    1    28143           0  ...                0               7000       0
1    2    19244           0  ...                0               6968       0
2    3    41354           0  ...                0               7034       0
3    4    14776           0  ...                0               6952       0
4    5    97752           0  ...                4               6935       1

[5 rows x 12 columns]
```

Out[37]:

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
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

3999 rows × 12 columns

In [38]:

```
print(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
None
```

In [39]:

```
df1 = df.drop(['ID#'],axis=1)
```

In [40]:

```
array=df1.values
array
```

Out[40]:

```
array([[28143,     0,     1, ...,     0,  7000,     0],
       [19244,     0,     1, ...,     0,  6968,     0],
       [41354,     0,     1, ...,     0,  7034,     0],
       ...,
       [73597,     0,     3, ...,     0,  1402,     1],
       [54899,     0,     1, ...,     1,  1401,     0],
       [ 3016,     0,     1, ...,     0,  1398,     0]])
```

In [41]:

```
stscaler = StandardScaler().fit(array)
X = stscaler.transform(array)
X
```

Out[41]:

```
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 [42]:

```
dbscan = DBSCAN(eps=0.95, min_samples=5)
dbscan.fit(X)
```

Out[42]:

```
DBSCAN(eps=0.95)
```

In [43]:

```
dbscan.labels_
```

Out[43]:

```
array([0, 0, 0, ..., 1, 0, 0])
```

In [44]:

```
cl=pd.DataFrame(dbscan.labels_,columns=['cluster'])
```

In [45]:

```
cl
pd.set_option("display.max_rows", None)
```

In [46]:

```
cl
```

Out[46]:

	cluster
0	0
1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	-1
9	1
10	0

In [47]:

```
df1 = pd.concat([df,cl],axis=1)
df1
```

Out[47]:

	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	
5	6	16420	0	1	1	1	0	0	
6	7	84914	0	3	1	1	27482	25	
7	8	20856	0	1	1	1	5250	4	
8	9	443003	0	3	2	1	1753	43	
9	10	104860	0	3	1	1	28426	28	

In [48]:

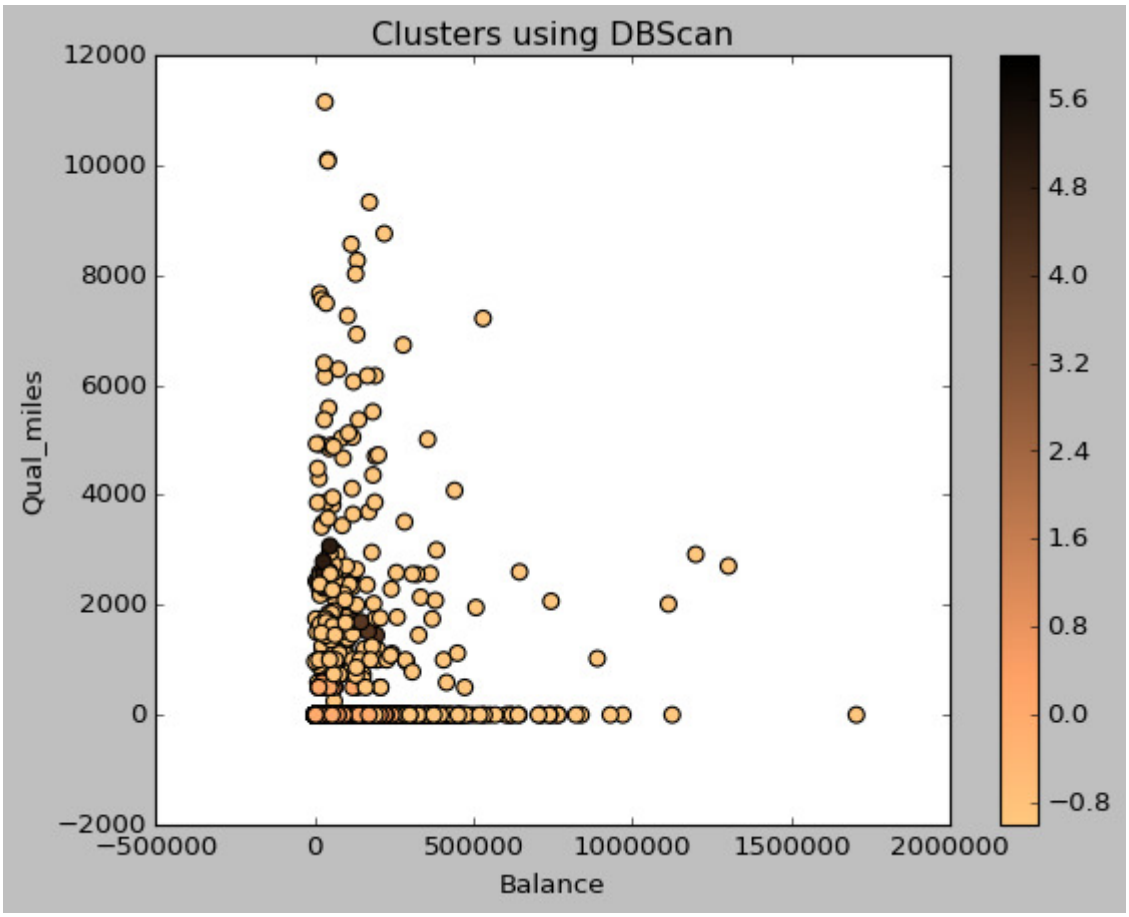
```
import matplotlib.pyplot as plt
>>> plt.style.use('classic')
```

In [49]:

```
df1.plot(x="Balance",y ="Qual_miles",c=dbscan.labels_ ,kind="scatter",s=50 ,cmap=plt.cm.cop
plt.title('Clusters using DBScan')
```

Out[49]:

```
Text(0.5, 1.0, 'Clusters using DBScan')
```

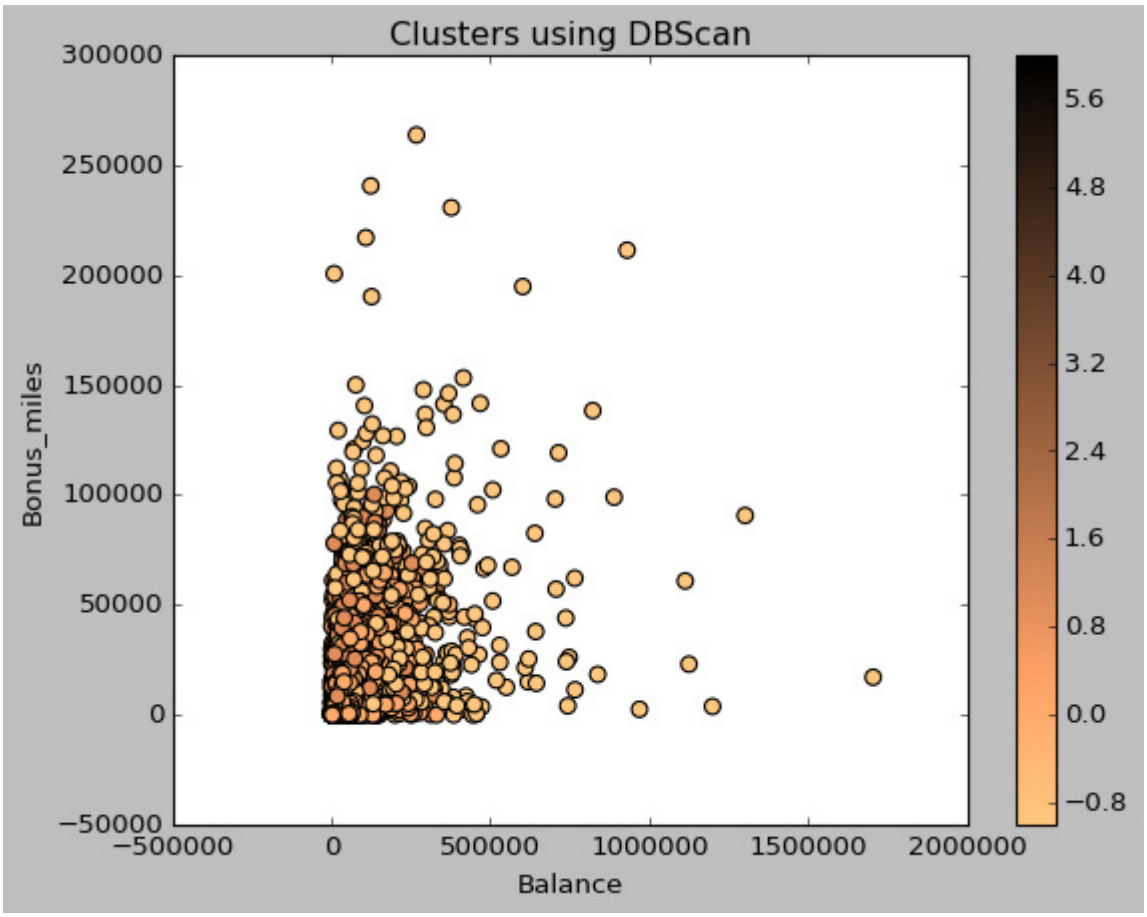


In [51]:

```
df1.plot(x="Balance",y ="Bonus_miles",c=dbscan.labels_ ,kind="scatter",s=50 ,cmap=plt.cm.co
plt.title('Clusters using DBScan')
```

Out[51]:

```
Text(0.5, 1.0, 'Clusters using DBScan')
```



In [52]:

```
dl = dbscan.labels_
```

In [53]:

```
from sklearn import metrics
```

In [54]:

```
import sklearn
sklearn.metrics.silhouette_score(X, dl)
```

Out[54]:

```
0.18564846327937293
```

In [55]:

```
from sklearn import metrics
```

In [56]:

```
from sklearn.cluster import KMeans
clf = KMeans(n_clusters=3)
y_kmeans = clf.fit_predict(X)
```

In [57]:

```
y_kmeans
```

Out[57]:

```
array([1, 1, 1, ..., 0, 1, 1], dtype=int32)
```

In [58]:

```
cl1=pd.DataFrame(y_kmeans,columns=['Kcluster'])
cl1
```

Out[58]:

	Kcluster
0	1
1	1
2	1
3	1
4	0
5	1
6	0
7	1
8	2
9	0
10	1

In [59]:

```
df2 = pd.concat([df1,cl1],axis=1)
df2
```

Out[59]:

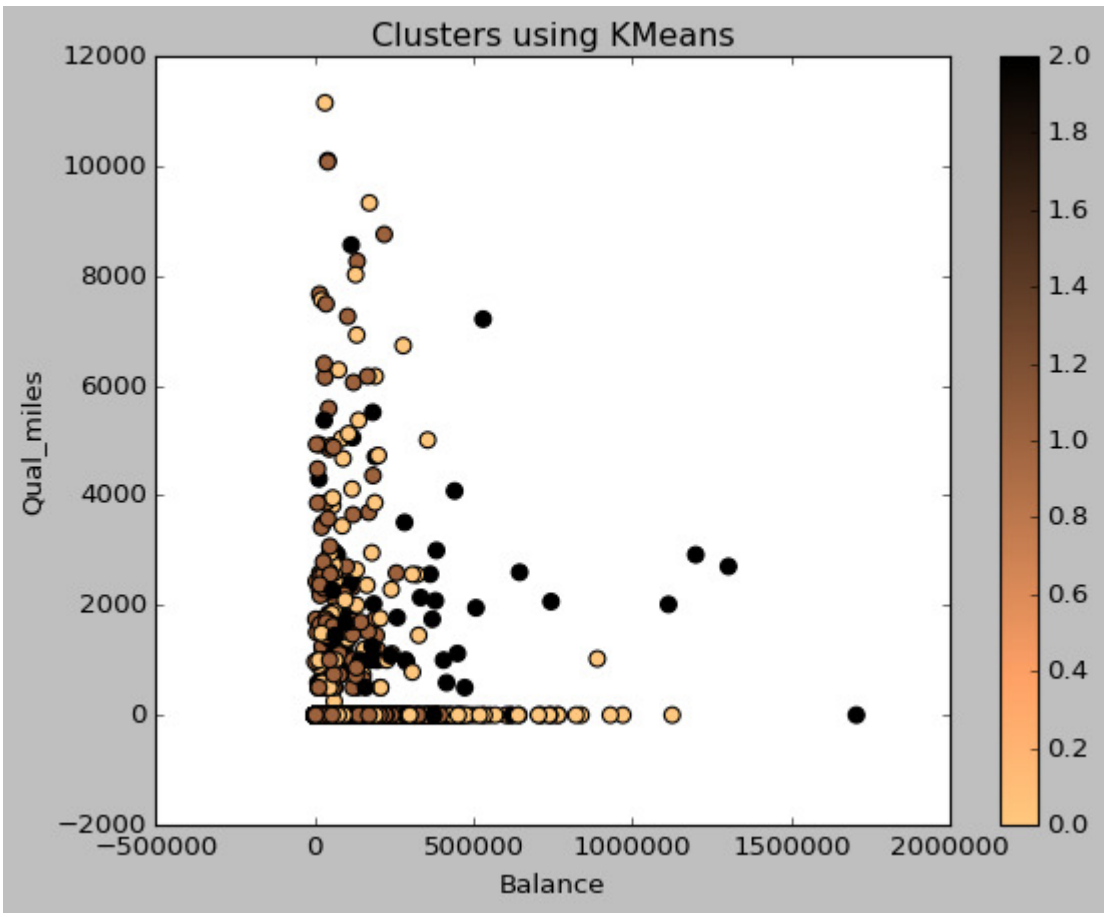
	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Flight_miles_
0	1	28143	0	1	1	1	174	1	
1	2	19244	0	1	1	1	215	2	
2	3	41354	0	1	1	1	4123	4	
3	4	14776	0	1	1	1	500	1	
4	5	97752	0	4	1	1	43300	26	
5	6	16420	0	1	1	1	0	0	
6	7	84914	0	3	1	1	27482	25	
7	8	20856	0	1	1	1	5250	4	
8	9	443003	0	3	2	1	1753	43	
9	10	104860	0	3	1	1	28426	28	

In [61]:

```
df2.plot(x="Balance",y ="Qual_miles",c=y_kmeans ,kind="scatter",s=50 ,cmap=plt.cm.copper_r)
plt.title('Clusters using KMeans')
```

Out[61]:

```
Text(0.5, 1.0, 'Clusters using KMeans')
```

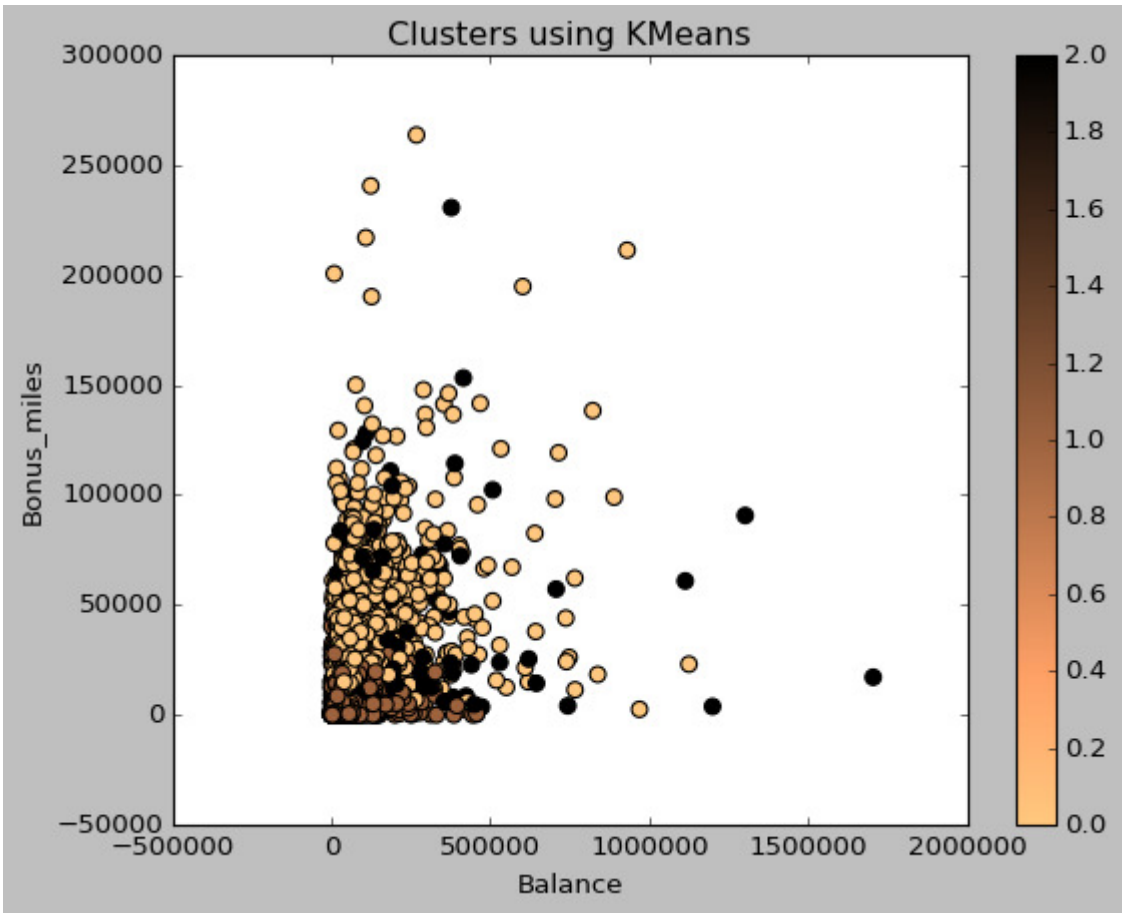


In [62]:

```
df2.plot(x="Balance",y ="Bonus_miles",c=y_kmeans ,kind="scatter",s=50 ,cmap=plt.cm.copper_r
plt.title('Clusters using KMeans')
```

Out[62]:

```
Text(0.5, 1.0, 'Clusters using KMeans')
```



In [63]:

```
sklearn.metrics.silhouette_score(X, y_kmeans)
```

Out[63]:

```
0.31191384766627117
```