K-Mean-Clustering

In [1]:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
```

In [2]:

```
univ=pd.read_csv("C:/Users/Ashraf/Documents/Datafiles/Universities.csv")
```

In [3]:

```
univ
```

Out[3]:

	Univ	SAT	Top10	Accept	SFRatio	Expenses	GradRate
0	Brown	1310	89	22	13	22704	94
1	CalTech	1415	100	25	6	63575	81
2	CMU	1260	62	59	9	25026	72
3	Columbia	1310	76	24	12	31510	88
4	Cornell	1280	83	33	13	21864	90
5	Dartmouth	1340	89	23	10	32162	95
6	Duke	1315	90	30	12	31585	95
7	Georgetown	1255	74	24	12	20126	92
8	Harvard	1400	91	14	11	39525	97
9	JohnsHopkins	1305	75	44	7	58691	87
10	MIT	1380	94	30	10	34870	91
11	Northwestern	1260	85	39	11	28052	89
12	NotreDame	1255	81	42	13	15122	94
13	PennState	1081	38	54	18	10185	80
14	Princeton	1375	91	14	8	30220	95
15	Purdue	1005	28	90	19	9066	69
16	Stanford	1360	90	20	12	36450	93
17	TexasA&M	1075	49	67	25	8704	67
18	UCBerkeley	1240	95	40	17	15140	78
19	UChicago	1290	75	50	13	38380	87
20	UMichigan	1180	65	68	16	15470	85
21	UPenn	1285	80	36	11	27553	90
22	UVA	1225	77	44	14	13349	92
23	UWisconsin	1085	40	69	15	11857	71
24	Yale	1375	95	19	11	43514	96

In [4]:

```
# Normalization function
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_univ_df = scaler.fit_transform(univ.iloc[:,1:])
```

In [5]:

```
scaled_univ_df
```

Out[5]:

```
array([[ 0.41028362,  0.6575195 , -0.88986682,  0.07026045, -0.33141256,
         0.82030265],
       [ 1.39925928,  1.23521235, -0.73465749, -1.68625071,  2.56038138,
        -0.64452351],
       [-0.06065717, -0.76045386,  1.02438157, -0.93346022, -0.16712136,
        -1.65863393],
       [ 0.41028362, -0.02520842, -0.78639393, -0.18066972,  0.29164871,
         0.14422904],
       [ 0.12771914,  0.34241431, -0.32076595,  0.07026045, -0.39084607,
         0.36958691],
       [ 0.69284809,  0.6575195 , -0.83813038, -0.68253005,  0.33778044,
         0.93298158],
       [ 0.4573777 ,  0.71003703, -0.47597528, -0.18066972,  0.29695528,
         0.93298158],
       [-0.10775125, -0.13024348, -0.78639393, -0.18066972, -0.51381683,
         0.59494478],
       [ 1.25797704,  0.76255456, -1.30375836, -0.43159988,  0.85874344,
         1.15833946],
       [ 0.36318954, -0.07772595,  0.24833493, -1.43532055,  2.21481798,
         0.0315501 ],
       [ 1.06960072,  0.92010716, -0.47597528, -0.68253005,  0.52938275,
         0.48226584],
       [-0.06065717,  0.44744937, -0.01034729, -0.43159988,  0.04698077,
         0.25690797],
       [-0.10775125,  0.23737924,  0.14486204,  0.07026045, -0.86787073,
         0.82030265],
       [-1.7466252 , -2.02087462,  0.76569936,  1.32491127, -1.21718409,
        -0.75720245],
       [ 1.02250664,  0.76255456, -1.30375836, -1.18439038,  0.20037583,
         0.93298158],
       [-2.46245521, -2.54604994,  2.6282113 ,  1.57584144, -1.29635802,
        -1.99667073],
       [ 0.88122441,  0.71003703, -0.9933397 , -0.18066972,  0.64117435,
         0.70762371],
       [-1.8031381 , -1.44318177,  1.43827311,  3.08142243, -1.32197103,
        -2.22202861],
       [-0.24903349,  0.97262469,  0.04138915,  1.07398111, -0.86659715,
        -0.98256032],
       [ 0.2219073 , -0.07772595,  0.55875358,  0.07026045,  0.77772991,
         0.0315501 ],
       [-0.81416244, -0.60290126,  1.49000956,  0.82305094, -0.84324827,
        -0.19380777],
       [ 0.17481322,  0.18486171, -0.16555662, -0.43159988,  0.01167444,
         0.36958691],
       [-0.39031573,  0.02730912,  0.24833493,  0.32119061, -0.99331788,
         0.59494478],
       [-1.70894994, -1.91583956,  1.541746  ,  0.57212078, -1.09888311,
        -1.77131286],
       [ 1.02250664,  0.97262469, -1.04507615, -0.43159988,  1.14098185,
         1.04566052]])
```

In [6]:

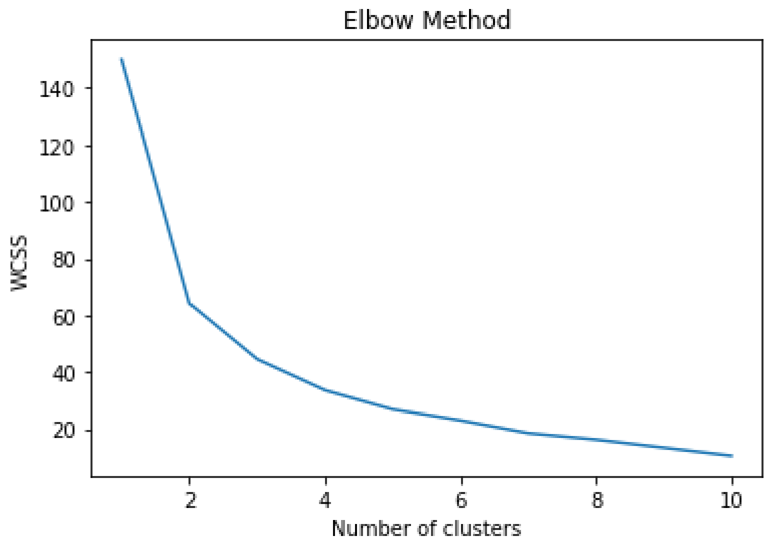
```
# How to find optimum number of  cluster
#The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluste
```

In [7]:

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

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

```
C:\Users\Ashraf\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881:
UserWarning: 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 th
e environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [8]:

```
#Build Cluster algorithm
from sklearn.cluster import KMeans
clusters_new = KMeans(4, random_state=42)
clusters_new.fit(scaled_univ_df)
```

Out[8]:

```
KMeans(n_clusters=4, random_state=42)
```

In [9]:

```
clusters_new.labels_
```

Out[9]:

```
array([2, 3, 0, 2, 0, 2, 2, 0, 2, 3, 2, 0, 0, 1, 2, 1, 2, 1, 0, 0, 0, 0,
       0, 1, 2])
```

In [10]:

```
#Assign clusters to the data set
univ['clusterid_new'] = clusters_new.labels_
```

In [11]:

```
#these are standardized values.
clusters_new.cluster_centers_
```

Out[11]:

```
array([[-0.12658888,  0.06407139,  0.2224667 ,  0.04516743, -0.38064332,
         0.02028221],
       [-1.93029211, -1.98148647,  1.59348244,  1.63857398, -1.23359906,
        -1.68680366],
       [ 0.80273428,  0.68086062, -0.90136381, -0.43159988,  0.44062556,
         0.79526289],
       [ 0.88122441,  0.5787432 , -0.24316128, -1.56078563,  2.38759968,
        -0.3064867 ]])
```

In [12]:

```
univ.groupby('clusterid_new').agg(['mean']).reset_index()
```

Out[12]:

	clusterid_new	SAT	Top10	Accept	SFRatio	Expenses	GradRate
		mean	mean	mean	mean	mean	mean
0	0	1253.000000	77.700000	43.500000	12.90	22008.200000	86.900000
1	1	1061.500000	38.750000	70.000000	19.25	9953.000000	71.750000
2	2	1351.666667	89.444444	21.777778	11.00	33615.555556	93.777778
3	3	1360.000000	87.500000	34.500000	6.50	61133.000000	84.000000

In [13]:

```
univ
```

Out[13]:

	Univ	SAT	Top10	Accept	SFRatio	Expenses	GradRate	clusterid_new
0	Brown	1310	89	22	13	22704	94	2
1	CalTech	1415	100	25	6	63575	81	3
2	CMU	1260	62	59	9	25026	72	0
3	Columbia	1310	76	24	12	31510	88	2
4	Cornell	1280	83	33	13	21864	90	0
5	Dartmouth	1340	89	23	10	32162	95	2
6	Duke	1315	90	30	12	31585	95	2
7	Georgetown	1255	74	24	12	20126	92	0
8	Harvard	1400	91	14	11	39525	97	2
9	JohnsHopkins	1305	75	44	7	58691	87	3
10	MIT	1380	94	30	10	34870	91	2
11	Northwestern	1260	85	39	11	28052	89	0
12	NotreDame	1255	81	42	13	15122	94	0
13	PennState	1081	38	54	18	10185	80	1
14	Princeton	1375	91	14	8	30220	95	2
15	Purdue	1005	28	90	19	9066	69	1
16	Stanford	1360	90	20	12	36450	93	2
17	TexasA&M	1075	49	67	25	8704	67	1
18	UCBerkeley	1240	95	40	17	15140	78	0
19	UChicago	1290	75	50	13	38380	87	0
20	UMichigan	1180	65	68	16	15470	85	0
21	UPenn	1285	80	36	11	27553	90	0
22	UVA	1225	77	44	14	13349	92	0
23	UWisconsin	1085	40	69	15	11857	71	1
24	Yale	1375	95	19	11	43514	96	2

In [ ]:

```

```