Perform hierarchical clustering (single linkage) and k-means clustering on the TopUniversities data using Scikit-Learn. Complete both clustering tasks in one program file.

a. Draw the dendrogram for hierarchical clustering.

b. Perform hierarchical clustering (with single linkage) and k-means clustering with 2 clusters. Compare the results of the two algorithms by displaying the cluster labels for each record on a table.

c.Let the range of the number of clusters be k = 2, 3, …, 15. Find the best k from this range for hierarchical clustering based on the silhouette score. Find the best k for k-means clustering based on the silhouette score and inertia (SSE) score.

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import style
from matplotlib import pyplot as plt
#import graphviz as gr
%matplotlib inline
style.use("fivethirtyeight")
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')
import warnings
warnings.filterwarnings('ignore')
pd.set_option("display.max_columns", 60)
pd.set_option('display.max_rows', 50)
pd.set_option('display.width', 1000)
```

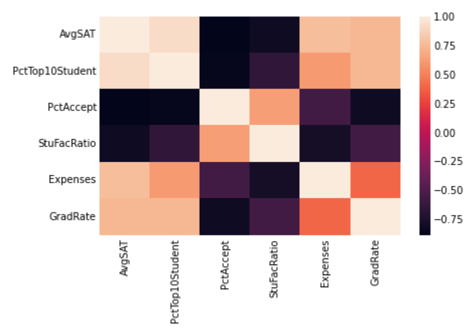
```
Mounted at /content/drive
```

In [2]:
```
import seaborn as sns
from sklearn.model_selection import train_test_split

df=pd.read_csv('/content/drive/MyDrive/DataMining/Files/TopUniversities.csv')

#see the corr matrix
cm=df.corr()
sns.heatmap(cm)
```

Out[2]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f663c79a2d0>
```



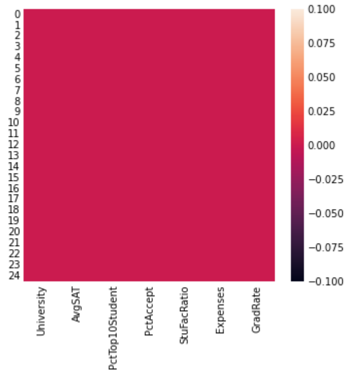
Not much Correlation present in the dataset

In [3]:
```
df.head
```

Out[3]:
```
<bound method NDFrame.head of         University  AvgSAT  PctTop10Student  PctAccept  StuFacRatio  Expenses  GradRate
0          Harvard   14.00          91          14          11    39.525        97
1        Princeton   13.75          91          14           8    30.220        95
2             Yale   13.75          95          19          11    43.514        96
3         Stanford   13.60          90          20          12    36.450        93
4              MIT   13.80          94          30          10    34.870        91
5             Duke   13.15          90          30          12    31.585        95
6          CalTech   14.15         100          25           6    63.575        81
7        Dartmouth   13.40          89          23          10    32.162        95
8            Brown   13.10          89          22          13    22.704        94
9     JohnsHopkins   13.05          75          44           7    58.691        87
10        UChicago   12.90          75          50          13    38.380        87
11           UPenn   12.85          80          36          11    27.553        90
12         Cornell   12.80          83          33          13    21.864        90
13    Northwestern   12.60          85          39          11    28.052        89
14        Columbia   13.10          76          24          12    31.510        88
15       NotreDame   12.55          81          42          13    15.122        94
16            UVir   12.25          77          44          14    13.349        92
17      Georgetown   12.55          74          24          12    20.126        92
18  CarnegieMellon   12.60          62          59           9    25.026        72
19       UMichigan   11.80          65          68          16    15.470        85
20      UCBerkeley   12.40          95          40          17    15.140        78
21      UWisconsin   10.85          40          69          15    11.857        71
22       PennState   10.81          38          54          18    10.185        80
23          Purdue   10.05          28          90          19     9.066        69
24        TexasA&M   10.75          49          67          25     8.704        67>
```

In [4]:
```
#Hitmap to check the null or drop columns
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5))
sns.heatmap(df.isnull()) ##to visualize the missing values
df.isnull().sum()
```

Out[4]:
```
University        0
AvgSAT            0
PctTop10Student   0
PctAccept         0
StuFacRatio       0
Expenses          0
GradRate          0
dtype: int64
```



No null values present in the dataset

In [5]:
```
df
```

Out[5]:

	University	AvgSAT	PctTop10Student	PctAccept	StuFacRatio	Expenses	GradRate
0	Harvard	14.00	91	14	11	39.525	97
1	Princeton	13.75	91	14	8	30.220	95
2	Yale	13.75	95	19	11	43.514	96
3	Stanford	13.60	90	20	12	36.450	93
4	MIT	13.80	94	30	10	34.870	91
5	Duke	13.15	90	30	12	31.585	95
6	CalTech	14.15	100	25	6	63.575	81
7	Dartmouth	13.40	89	23	10	32.162	95
8	Brown	13.10	89	22	13	22.704	94
9	JohnsHopkins	13.05	75	44	7	58.691	87
10	UChicago	12.90	75	50	13	38.380	87
11	UPenn	12.85	80	36	11	27.553	90
12	Cornell	12.80	83	33	13	21.864	90
13	Northwestern	12.60	85	39	11	28.052	89
14	Columbia	13.10	76	24	12	31.510	88
15	NotreDame	12.55	81	42	13	15.122	94
16	UVir	12.25	77	44	14	13.349	92
17	Georgetown	12.55	74	24	12	20.126	92
18	CarnegieMellon	12.60	62	59	9	25.026	72
19	UMichigan	11.80	65	68	16	15.470	85
20	UCBerkeley	12.40	95	40	17	15.140	78
21	UWisconsin	10.85	40	69	15	11.857	71
22	PennState	10.81	38	54	18	10.185	80
23	Purdue	10.05	28	90	19	9.066	69
24	TexasA&M	10.75	49	67	25	8.704	67

Hierarchical Clustering

In [6]:
```
#dropping the university name column
df = df.drop (['University'], axis='columns')
df
```

Out[6]:

	AvgSAT	PctTop10Student	PctAccept	StuFacRatio	Expenses	GradRate
0	14.00	91	14	11	39.525	97
1	13.75	91	14	8	30.220	95
2	13.75	95	19	11	43.514	96
3	13.60	90	20	12	36.450	93
4	13.80	94	30	10	34.870	91
5	13.15	90	30	12	31.585	95
6	14.15	100	25	6	63.575	81
7	13.40	89	23	10	32.162	95
8	13.10	89	22	13	22.704	94
9	13.05	75	44	7	58.691	87
10	12.90	75	50	13	38.380	87
11	12.85	80	36	11	27.553	90
12	12.80	83	33	13	21.864	90
13	12.60	85	39	11	28.052	89
14	13.10	76	24	12	31.510	88
15	12.55	81	42	13	15.122	94
16	12.25	77	44	14	13.349	92
17	12.55	74	24	12	20.126	92
18	12.60	62	59	9	25.026	72
19	11.80	65	68	16	15.470	85
20	12.40	95	40	17	15.140	78
21	10.85	40	69	15	11.857	71
22	10.81	38	54	18	10.185	80
23	10.05	28	90	19	9.066	69
24	10.75	49	67	25	8.704	67

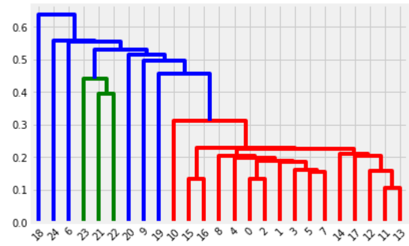
In [7]:
```
# Normalize numeric features and encode categorical features to dummies (by if-else)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
numary = scaler.fit_transform(df[['AvgSAT','PctTop10Student','PctAccept','StuFacRatio','Expenses','GradRate']])
normdf = pd.DataFrame(data=numary, columns=['AvgSAT','PctTop10Student','PctAccept','StuFacRatio','Expenses','GradRate'])
normdf
```

Out[7]:

	AvgSAT	PctTop10Student	PctAccept	StuFacRatio	Expenses	GradRate
0	0.963415	0.875000	0.000000	0.263158	0.561699	1.000000
1	0.902439	0.875000	0.000000	0.105263	0.392120	0.933333
2	0.902439	0.930556	0.065789	0.263158	0.634397	0.966667
3	0.865854	0.861111	0.078947	0.315789	0.505659	0.866667
4	0.914634	0.916667	0.210526	0.210526	0.476864	0.800000
5	0.756098	0.861111	0.210526	0.315789	0.416996	0.933333
6	1.000000	1.000000	0.144737	0.000000	1.000000	0.466667
7	0.817073	0.847222	0.118421	0.210526	0.427512	0.933333
8	0.743902	0.847222	0.105263	0.368421	0.255144	0.900000
9	0.731707	0.652778	0.394737	0.052632	0.910991	0.666667
10	0.695122	0.652778	0.473684	0.368421	0.540832	0.666667
11	0.682927	0.722222	0.289474	0.263158	0.343515	0.766667
12	0.670732	0.763889	0.250000	0.368421	0.239835	0.766667
13	0.621951	0.791667	0.328947	0.263158	0.352609	0.733333
14	0.743902	0.666667	0.131579	0.315789	0.415629	0.700000
15	0.609756	0.736111	0.368421	0.368421	0.116965	0.900000
16	0.536585	0.680556	0.394737	0.421053	0.084653	0.833333
17	0.609756	0.638889	0.131579	0.315789	0.208161	0.833333
18	0.621951	0.472222	0.592105	0.157895	0.297461	0.166667
19	0.426829	0.513889	0.710526	0.526316	0.123307	0.600000
20	0.573171	0.930556	0.342105	0.578947	0.117293	0.366667
21	0.195122	0.166667	0.723684	0.473684	0.057462	0.133333
22	0.185366	0.138889	0.526316	0.631579	0.026991	0.433333
23	0.000000	0.000000	1.000000	0.684211	0.006597	0.066667
24	0.170732	0.291667	0.697368	1.000000	0.000000	0.000000

a. Draw the dendrogram for hierarchical clustering.

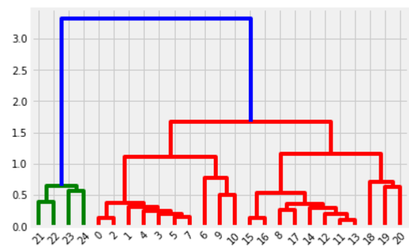
In [8]:
```
# Draw dendrogram
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(normdf, method='single'))
```



Doing some additional exploration as discussed in class

In [9]:
```
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
```

In [10]:
```
#Creating a new dendogram using 'Ward' method
dendrogram1 = sch.dendrogram(sch.linkage(normdf, method='ward'))
```



We can see how this method has created different clusters compared to the previous single linkage method. Here we can clearly see 3 broad clusters however potentially they have joined and can be treated as two clusters

b. Perform hierarchical clustering (with single linkage) and k-means clustering with 2 clusters. Compare the results of the two algorithms by displaying the cluster labels for each record on a table.

In [14]:
```
# Single linkage hierarchical (agglomerative) clustering
from sklearn.cluster import AgglomerativeClustering
agg = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='single')
pred = agg.fit_predict(normdf)
normdf['Cluster'] = pred
normdf
```

Out[14]:

	AvgSAT	PctTop10Student	PctAccept	StuFacRatio	Expenses	GradRate	Cluster
0	0.963415	0.875000	0.000000	0.263158	0.561699	1.000000	0
1	0.902439	0.875000	0.000000	0.105263	0.392120	0.933333	0
2	0.902439	0.930556	0.065789	0.263158	0.634397	0.966667	0
3	0.865854	0.861111	0.078947	0.315789	0.505659	0.866667	0
4	0.914634	0.916667	0.210526	0.210526	0.476864	0.800000	0
5	0.756098	0.861111	0.210526	0.315789	0.416996	0.933333	0
6	1.000000	1.000000	0.144737	0.000000	1.000000	0.466667	0
7	0.817073	0.847222	0.118421	0.210526	0.427512	0.933333	0
8	0.743902	0.847222	0.105263	0.368421	0.255144	0.900000	0
9	0.731707	0.652778	0.394737	0.052632	0.910991	0.666667	0
10	0.695122	0.652778	0.473684	0.368421	0.540832	0.666667	0
11	0.682927	0.722222	0.289474	0.263158	0.343515	0.766667	0
12	0.670732	0.763889	0.250000	0.368421	0.239835	0.766667	0
13	0.621951	0.791667	0.328947	0.263158	0.352609	0.733333	0
14	0.743902	0.666667	0.131579	0.315789	0.415629	0.700000	0
15	0.609756	0.736111	0.368421	0.368421	0.116965	0.900000	0
16	0.536585	0.680556	0.394737	0.421053	0.084653	0.833333	0
17	0.609756	0.638889	0.131579	0.315789	0.208161	0.833333	0
18	0.621951	0.472222	0.592105	0.157895	0.297461	0.166667	1
19	0.426829	0.513889	0.710526	0.526316	0.123307	0.600000	0
20	0.573171	0.930556	0.342105	0.578947	0.117293	0.366667	0
21	0.195122	0.166667	0.723684	0.473684	0.057462	0.133333	0
22	0.185366	0.138889	0.526316	0.631579	0.026991	0.433333	0
23	0.000000	0.000000	1.000000	0.684211	0.006597	0.066667	0
24	0.170732	0.291667	0.697368	1.000000	0.000000	0.000000	0

In [15]:
```
df['Cluster'] = pred
df
```

Out[15]:

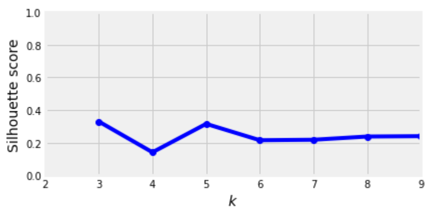
	AvgSAT	PctTop10Student	PctAccept	StuFacRatio	Expenses	GradRate	Cluster
0	14.00	91	14	11	39.525	97	0
1	13.75	91	14	8	30.220	95	0
2	13.75	95	19	11	43.514	96	0
3	13.60	90	20	12	36.450	93	0
4	13.80	94	30	10	34.870	91	0
5	13.15	90	30	12	31.585	95	0
6	14.15	100	25	6	63.575	81	0
7	13.40	89	23	10	32.162	95	0
8	13.10	89	22	13	22.704	94	0
9	13.05	75	44	7	58.691	87	0
10	12.90	75	50	13	38.380	87	0
11	12.85	80	36	11	27.553	90	0
12	12.80	83	33	13	21.864	90	0
13	12.60	85	39	11	28.052	89	0
14	13.10	76	24	12	31.510	88	0
15	12.55	81	42	13	15.122	94	0
16	12.25	77	44	14	13.349	92	0
17	12.55	74	24	12	20.126	92	0
18	12.60	62	59	9	25.026	72	1
19	11.80	65	68	16	15.470	85	0
20	12.40	95	40	17	15.140	78	0
21	10.85	40	69	15	11.857	71	0
22	10.81	38	54	18	10.185	80	0
23	10.05	28	90	19	9.066	69	0
24	10.75	49	67	25	8.704	67	0

c.Let the range of the number of clusters be k = 2, 3, …, 15. Find the best k from this range for hierarchical clustering based on the silhouette score. Find the best k for k-means clustering based on the silhouette score and inertia (SSE) score.

In [16]:
```
# Find the best number of clusters (k) based on silhouette score
from sklearn.metrics import silhouette_score
agg_per_k = [AgglomerativeClustering(n_clusters=k, linkage='single').fit(normdf)
             for k in range(2, 15)]
silhouette_scores = [silhouette_score(normdf, model.labels_)
for model in agg_per_k[1:]]
silhouette_scores
```

Out[16]:
```
[0.3298692358328648,
 0.14103419528584418,
 0.3153455351816817,
 0.21503619932634635,
 0.2179131096401342,
 0.237703475188227,
 0.23996672027224084,
 0.2256135863801685,
 0.12955286995839427,
 0.16082507160418927,
 0.20848368075223134,
 0.16126376721319005]
```

In [17]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(6, 3))
plt.plot(range(3, 15), silhouette_scores, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Silhouette score", fontsize=14)
plt.axis([2, 9, 0, 1])
plt.show()
# silhouette_score is the largest when k=2
```



Silhouette_score is the largest when k=3. This means we can do 3 clusters for this example

K Means Clustering with 2 clusters

In [18]:
```
# k-Means clustering
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=1)
pred = kmeans.fit_predict(normdf)
normdf['Cluster'] = pred
normdf
```

Out[18]:

	AvgSAT	PctTop10Student	PctAccept	StuFacRatio	Expenses	GradRate	Cluster
0	0.963415	0.875000	0.000000	0.263158	0.561699	1.000000	0
1	0.902439	0.875000	0.000000	0.105263	0.392120	0.933333	0
2	0.902439	0.930556	0.065789	0.263158	0.634397	0.966667	0
3	0.865854	0.861111	0.078947	0.315789	0.505659	0.866667	0
4	0.914634	0.916667	0.210526	0.210526	0.476864	0.800000	0
5	0.756098	0.861111	0.210526	0.315789	0.416996	0.933333	0
6	1.000000	1.000000	0.144737	0.000000	1.000000	0.466667	0
7	0.817073	0.847222	0.118421	0.210526	0.427512	0.933333	0
8	0.743902	0.847222	0.105263	0.368421	0.255144	0.900000	0
9	0.731707	0.652778	0.394737	0.052632	0.910991	0.666667	0
10	0.695122	0.652778	0.473684	0.368421	0.540832	0.666667	0
11	0.682927	0.722222	0.289474	0.263158	0.343515	0.766667	0
12	0.670732	0.763889	0.250000	0.368421	0.239835	0.766667	0
13	0.621951	0.791667	0.328947	0.263158	0.352609	0.733333	0
14	0.743902	0.666667	0.131579	0.315789	0.415629	0.700000	0
15	0.609756	0.736111	0.368421	0.368421	0.116965	0.900000	0
16	0.536585	0.680556	0.394737	0.421053	0.084653	0.833333	0
17	0.609756	0.638889	0.131579	0.315789	0.208161	0.833333	0
18	0.621951	0.472222	0.592105	0.157895	0.297461	0.166667	1
19	0.426829	0.513889	0.710526	0.526316	0.123307	0.600000	1
20	0.573171	0.930556	0.342105	0.578947	0.117293	0.366667	0
21	0.195122	0.166667	0.723684	0.473684	0.057462	0.133333	1
22	0.185366	0.138889	0.526316	0.631579	0.026991	0.433333	1
23	0.000000	0.000000	1.000000	0.684211	0.006597	0.066667	1
24	0.170732	0.291667	0.697368	1.000000	0.000000	0.000000	1

In [19]:
```
kmeans.cluster_centers_
```

Out[19]:
```
array([[0.75481386, 0.80263158, 0.21260388, 0.28254848, 0.42109867,
        0.79122807, 0.        ],
       [0.26666667, 0.26388889, 0.70833333, 0.57894737, 0.0853031 ,
        0.23333333, 0.16666667]])
```

In [20]:
```
df['Cluster'] = pred
df
```

Out[20]:

	AvgSAT	PctTop10Student	PctAccept	StuFacRatio	Expenses	GradRate	Cluster
0	14.00	91	14	11	39.525	97	0
1	13.75	91	14	8	30.220	95	0
2	13.75	95	19	11	43.514	96	0
3	13.60	90	20	12	36.450	93	0
4	13.80	94	30	10	34.870	91	0
5	13.15	90	30	12	31.585	95	0
6	14.15	100	25	6	63.575	81	0
7	13.40	89	23	10	32.162	95	0
8	13.10	89	22	13	22.704	94	0
9	13.05	75	44	7	58.691	87	0
10	12.90	75	50	13	38.380	87	0
11	12.85	80	36	11	27.553	90	0
12	12.80	83	33	13	21.864	90	0
13	12.60	85	39	11	28.052	89	0
14	13.10	76	24	12	31.510	88	0
15	12.55	81	42	13	15.122	94	0
16	12.25	77	44	14	13.349	92	0
17	12.55	74	24	12	20.126	92	0
18	12.60	62	59	9	25.026	72	1
19	11.80	65	68	16	15.470	85	1
20	12.40	95	40	17	15.140	78	0
21	10.85	40	69	15	11.857	71	1
22	10.81	38	54	18	10.185	80	1
23	10.05	28	90	19	9.066	69	1
24	10.75	49	67	25	8.704	67	1

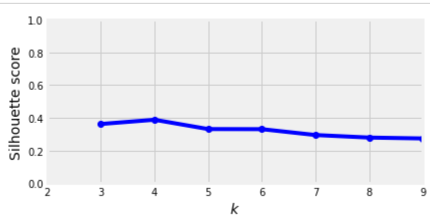
Let the range of the number of clusters be k = 2, 3, …, 15. Find the best k from this range for hierarchical clustering based on the silhouette score. Find the best k for k-means clustering based on the silhouette score and inertia (SSE) score.

In [21]:
```
# Find the best k based on silhouette score
from sklearn.metrics import silhouette_score
kmeans_per_k = [KMeans(n_clusters=k, random_state=1).fit(normdf)
                for k in range(2, 15)]
silhouette_scores = [silhouette_score(normdf, model.labels_)
                     for model in kmeans_per_k[1:]]
silhouette_scores
```

Out[21]:
```
[0.36224461353311166,
 0.38824797046932874,
 0.3312585950243881,
 0.33064779436187053,
 0.29478237159067766,
 0.2791879807333731,
 0.27377996211572,
 0.251587051791595,
 0.21987109324508988,
 0.20551795935301753,
 0.21058410197863467,
 0.1695178912597066]
```

silhouette_scores has improved slightly in K-Means clustering compared to hierarchical clustering

In [40]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(6, 3))
plt.plot(range(3, 15), silhouette_scores, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Silhouette score", fontsize=14)
plt.axis([2, 9, 0, 1])
plt.show()
```

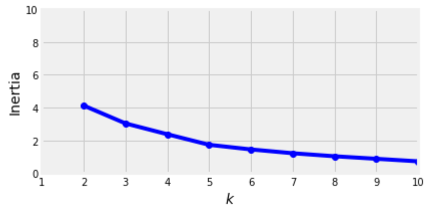


Silhouette_score is the largest when k=4

In [41]:
```
# Find the best k based on inertia (SSE) score
inertias = [model.inertia_ for model in kmeans_per_k]
inertias
```

Out[41]:
```
[4.123452442397178,
 3.033332821193251,
 2.3728787761000536,
 1.7285680842090172,
 1.445647696186281,
 1.213425909585292,
 1.0291133191249702,
 0.874105522476116,
 0.7191461832968294,
 0.5399459160057575,
 0.46210574681607375,
 0.35493170150747966,
 0.27747308171260643]
```

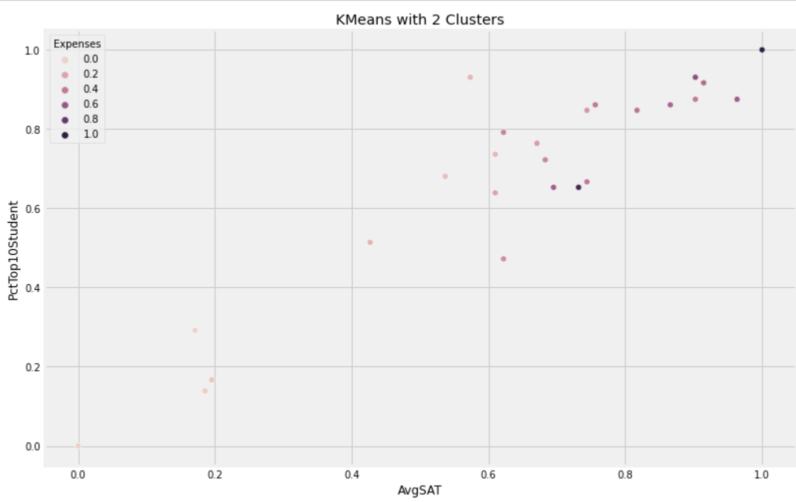
In [42]:
```
plt.figure(figsize=(6, 3))
plt.plot(range(2, 15), inertias, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Inertia", fontsize=14)
plt.axis([1, 10, 0, 10])
plt.show()
# elbow point occurs at k=3
```



Doing some additional visual work here

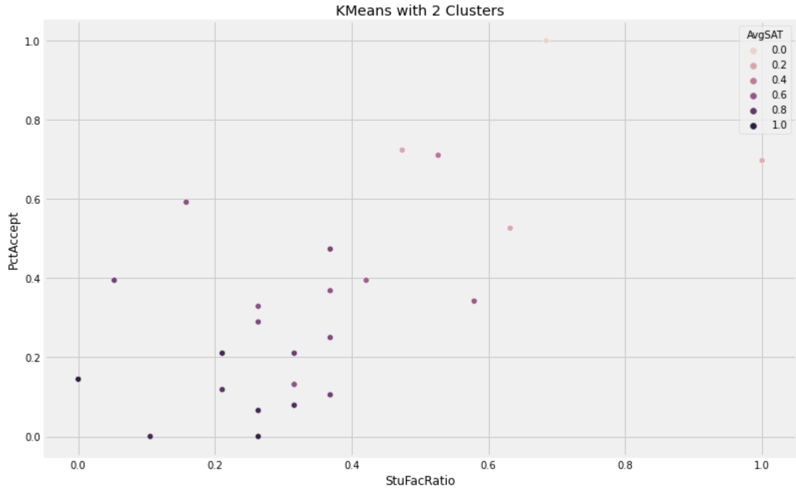
In [45]:
```
# 2 cluster plot
km3 = KMeans(n_clusters=2).fit(normdf)

normdf['Labels'] = km3.labels_
plt.figure(figsize=(12, 8))
sns.scatterplot(normdf['AvgSAT'], normdf['PctTop10Student'], hue=normdf['Expenses'],
                )
plt.title('KMeans with 2 Clusters')
plt.show()
```



In [46]:
```
# 2 cluster plot
km3 = KMeans(n_clusters=2).fit(normdf)

normdf['Labels'] = km3.labels_
plt.figure(figsize=(12, 8))
sns.scatterplot(normdf['StuFacRatio'], normdf['PctAccept'], hue=normdf['AvgSAT'],
                )
plt.title('KMeans with 2 Clusters')
plt.show()
```



Above illustration shows how clusters changed based on the parameters and similarly we can visually analyse different clusters.