In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv("C:/Users/USER/Desktop/Datasets/University_Clustering.csv")
```

In [4]:
```
del df["Univ"]

df.head(7)
```

Out[4]:

	State	SAT	Top10	Accept	SFRatio	Expenses	GradRate
0	RI	1310	89	22	13	22,704	94
1	CA	1415	100	25	6	63,575	81
2	PA	1260	62	59	9	25,026	72
3	NY	1310	76	24	12	31,510	88
4	NY	1280	83	33	13	21,864	90
5	NH	1340	89	23	10	32,162	95
6	NC	1315	90	30	12	31,585	95

In [5]:
```
df.columns
```

Out[5]:
```
Index(['State', 'SAT', 'Top10', 'Accept', 'SFRatio', 'Expenses', 'GradRate'], d
type='object')
```

In [6]:
```
df.shape
```

Out[6]:
```
(25, 7)
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   State     25 non-null     object
 1   SAT       25 non-null     int64
 2   Top10     25 non-null     int64
 3   Accept    25 non-null     int64
 4   SFRatio   25 non-null     int64
 5   Expenses  25 non-null     object
 6   GradRate  25 non-null     int64
dtypes: int64(5), object(2)
memory usage: 1.5+ KB
```

In [8]:
```
df['Expenses']=df['Expenses'].str.replace(',','').astype(int)
```

In [9]:
```
df
```

Out[9]:

	State	SAT	Top10	Accept	SFRatio	Expenses	GradRate
0	RI	1310	89	22	13	22704	94
1	CA	1415	100	25	6	63575	81
2	PA	1260	62	59	9	25026	72
3	NY	1310	76	24	12	31510	88
4	NY	1280	83	33	13	21864	90
5	NH	1340	89	23	10	32162	95
6	NC	1315	90	30	12	31585	95
7	DC	1255	74	24	12	20126	92
8	MA	1400	91	14	11	39525	97
9	MD	1305	75	44	7	58691	87
10	MA	1380	94	30	10	34870	91
11	IL	1260	85	39	11	28052	89
12	IN	1255	81	42	13	15122	94
13	PA	1081	38	54	18	10185	80
14	NJ	1375	91	14	8	30220	95
15	IN	1005	28	90	19	9066	69
16	CA	1360	90	20	12	36450	93
17	TX	1075	49	67	25	8704	67
18	CA	1240	95	40	17	15140	78
19	IL	1290	75	50	13	38380	87
20	MI	1180	65	68	16	15470	85
21	PA	1285	80	36	11	27553	90
22	VA	1225	77	44	14	13349	92
23	WI	1085	40	69	15	11857	71
24	CT	1375	95	19	11	43514	96

In [11]:
```
df_x= df.iloc[:,1:]
df_x
```

Out[11]:

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

In [15]:
```
df_x.columns
```

Out[15]:
```
Index(['SAT', 'Top10', 'Accept', 'SFRatio', 'Expenses', 'GradRate'], dtype='obj
ect')
```

In [18]:
```
df_x.dtypes
```

Out[18]:
```
SAT         int64
Top10       int64
Accept      int64
SFRatio     int64
Expenses    int32
GradRate    int64
dtype: object
```

In [20]:
```
df_x.describe()
```

Out[20]:

	SAT	Top10	Accept	SFRatio	Expenses	GradRate
count	25.000000	25.000000	25.000000	25.00000	25.000000	25.000000
mean	1266.440000	76.480000	39.200000	12.72000	27388.000000	86.720000
std	108.359771	19.433905	19.727308	4.06735	14424.883165	9.057778
min	1005.000000	28.000000	14.000000	6.00000	8704.000000	67.000000
25%	1240.000000	74.000000	24.000000	11.00000	15140.000000	81.000000
50%	1285.000000	81.000000	36.000000	12.00000	27553.000000	90.000000
75%	1340.000000	90.000000	50.000000	14.00000	34870.000000	94.000000
max	1415.000000	100.000000	90.000000	25.00000	63575.000000	97.000000

In [21]:
```
df_x.median()
```

Out[21]:
```
SAT          1285.0
Top10          81.0
Accept         36.0
SFRatio        12.0
Expenses    27553.0
GradRate       90.0
dtype: float64
```

In [ ]:
```
df_x.mode()
```

In [23]:
```
df_x.var()
```

Out[23]:
```
SAT         1.174184e+04
Top10       3.776767e+02
Accept      3.891667e+02
SFRatio     1.654333e+01
Expenses    2.080773e+08
GradRate    8.204333e+01
dtype: float64
```

In [24]:
```
df_x.skew()
```

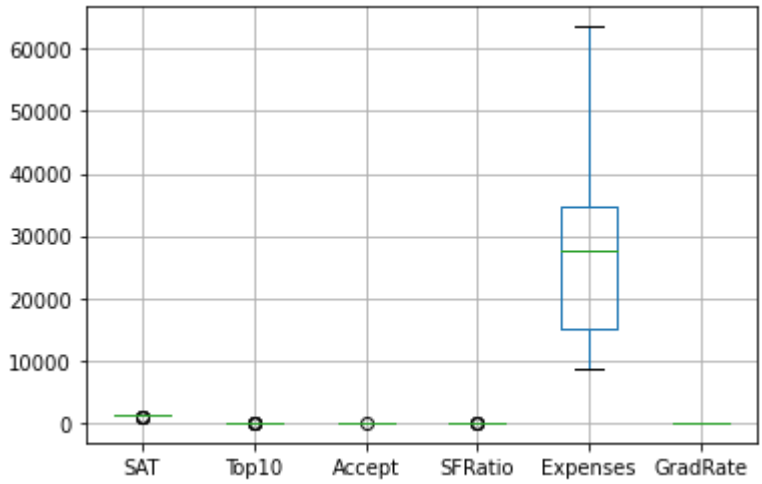
Out[24]:
```
SAT        -0.950910
Top10      -1.219756
Accept      0.867538
SFRatio     1.120719
Expenses    0.852350
GradRate   -0.999521
dtype: float64
```

In [25]:
```
df_x.kurtosis()
```

Out[25]:
```
SAT         0.355122
Top10       0.700307
Accept      0.252908
SFRatio     2.353743
Expenses    0.638162
GradRate   -0.164740
dtype: float64
```

In [26]:
```
boxplot=df_x.boxplot()
boxplot
```

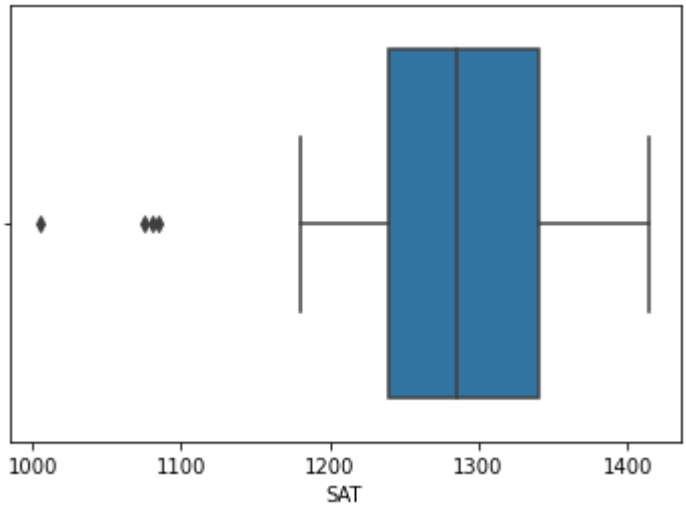
Out[26]:
```
<AxesSubplot:>
```



In [27]:
```
sns.boxplot(df_x["SAT"])
```

```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[27]: <AxesSubplot:xlabel='SAT'>

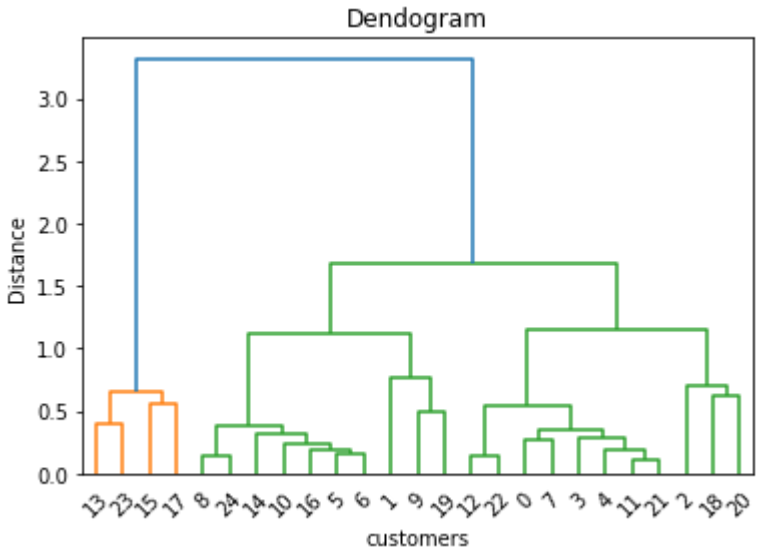


In [29]:
```
df_N=(df_x-df_x.min())/(df_x.max()-df_x.min())
df_N
```

Out[29]:

	SAT	Top10	Accept	SFRatio	Expenses	GradRate
0	0.743902	0.847222	0.105263	0.368421	0.255144	0.900000
1	1.000000	1.000000	0.144737	0.000000	1.000000	0.466667
2	0.621951	0.472222	0.592105	0.157895	0.297461	0.166667
3	0.743902	0.666667	0.131579	0.315789	0.415629	0.700000
4	0.670732	0.763889	0.250000	0.368421	0.239835	0.766667
5	0.817073	0.847222	0.118421	0.210526	0.427512	0.933333
6	0.756098	0.861111	0.210526	0.315789	0.416996	0.933333
7	0.609756	0.638889	0.131579	0.315789	0.208161	0.833333
8	0.963415	0.875000	0.000000	0.263158	0.561699	1.000000
9	0.731707	0.652778	0.394737	0.052632	0.910991	0.666667
10	0.914634	0.916667	0.210526	0.210526	0.476864	0.800000
11	0.621951	0.791667	0.328947	0.263158	0.352609	0.733333
12	0.609756	0.736111	0.368421	0.368421	0.116965	0.900000
13	0.185366	0.138889	0.526316	0.631579	0.026991	0.433333
14	0.902439	0.875000	0.000000	0.105263	0.392120	0.933333
15	0.000000	0.000000	1.000000	0.684211	0.006597	0.066667
16	0.865854	0.861111	0.078947	0.315789	0.505659	0.866667
17	0.170732	0.291667	0.697368	1.000000	0.000000	0.000000
18	0.573171	0.930556	0.342105	0.578947	0.117293	0.366667
19	0.695122	0.652778	0.473684	0.368421	0.540832	0.666667
20	0.426829	0.513889	0.710526	0.526316	0.123307	0.600000
21	0.682927	0.722222	0.289474	0.263158	0.343515	0.766667
22	0.536585	0.680556	0.394737	0.421053	0.084653	0.833333
23	0.195122	0.166667	0.723684	0.473684	0.057462	0.133333
24	0.902439	0.930556	0.065789	0.263158	0.634397	0.966667

In [30]:
```
import scipy.cluster.hierarchy as sch
dendogram = sch.dendrogram(sch.linkage(df_N,method = 'ward'))
plt.title('Dendogram')
plt.xlabel('customers')
plt.ylabel('Distance')
plt.show()
```



In [31]:
```
from  sklearn.cluster import AgglomerativeClustering
clf=AgglomerativeClustering(n_clusters=3,affinity='euclidean',linkage ='complete'
cluster=clf.fit_predict(df_N)
cluster
```

Out[31]:
```
array([0, 2, 0, 0, 0, 2, 2, 0, 2, 2, 2, 0, 0, 1, 2, 1, 2, 1, 0, 2, 0, 0,
       0, 1, 2], dtype=int64)
```

In [32]:
```
df["Clusters"]= pd.Series(cluster)
```

In [33]:
```
df
```

Out[33]:

	State	SAT	Top10	Accept	SFRatio	Expenses	GradRate	Clusters
0	RI	1310	89	22	13	22704	94	0
1	CA	1415	100	25	6	63575	81	2
2	PA	1260	62	59	9	25026	72	0
3	NY	1310	76	24	12	31510	88	0
4	NY	1280	83	33	13	21864	90	0
5	NH	1340	89	23	10	32162	95	2
6	NC	1315	90	30	12	31585	95	2
7	DC	1255	74	24	12	20126	92	0
8	MA	1400	91	14	11	39525	97	2
9	MD	1305	75	44	7	58691	87	2
10	MA	1380	94	30	10	34870	91	2
11	IL	1260	85	39	11	28052	89	0
12	IN	1255	81	42	13	15122	94	0
13	PA	1081	38	54	18	10185	80	1
14	NJ	1375	91	14	8	30220	95	2
15	IN	1005	28	90	19	9066	69	1
16	CA	1360	90	20	12	36450	93	2
17	TX	1075	49	67	25	8704	67	1
18	CA	1240	95	40	17	15140	78	0
19	IL	1290	75	50	13	38380	87	2
20	MI	1180	65	68	16	15470	85	0
21	PA	1285	80	36	11	27553	90	0
22	VA	1225	77	44	14	13349	92	0
23	WI	1085	40	69	15	11857	71	1
24	CT	1375	95	19	11	43514	96	2

In [ ]:
```

```