In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
```

In [2]:
```
data = pd.read_csv("wine.csv")
data.head()
```

Out[2]:

	Type	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflavanoids
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39

In [3]:
```
data.dtypes
```

Out[3]:
```
Type                int64
Alcohol           float64
Malic             float64
Ash               float64
Alcalinity        float64
Magnesium           int64
Phenols           float64
Flavanoids        float64
Nonflavanoids     float64
Proanthocyanins   float64
Color             float64
Hue               float64
Dilution          float64
Proline             int64
dtype: object
```

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
Type                0
Alcohol             0
Malic               0
Ash                 0
Alcalinity          0
Magnesium           0
Phenols             0
Flavanoids          0
Nonflavanoids       0
Proanthocyanins     0
Color               0
Hue                 0
Dilution            0
Proline             0
dtype: int64
```

In [5]:
```
data.duplicated().value_counts()
```

Out[5]:
```
False    178
dtype: int64
```

In [6]:
```
dataa = data.iloc[:,1:]
data_cluster = data.iloc[:,:1]
data_cluster.head()
```

Out[6]:

	Type
0	1
1	1
2	1
3	1
4	1

In [7]:
```
df = data.iloc[:,1:].values
```

In [8]:
```
df_norm = scale(df)
df_norm
```

Out[8]:
```
array([[ 1.51861254, -0.5622498 ,  0.23205254, ...,  0.36217728,
         1.84791957,  1.01300893],
       [ 0.24628963, -0.49941338, -0.82799632, ...,  0.40605066,
         1.1134493 ,  0.96524152],
       [ 0.19687903,  0.02123125,  1.10933436, ...,  0.31830389,
         0.78858745,  1.39514818],
       ...,
       [ 0.33275817,  1.74474449, -0.38935541, ..., -1.61212515,
        -1.48544548,  0.28057537],
       [ 0.20923168,  0.22769377,  0.01273209, ..., -1.56825176,
        -1.40069891,  0.29649784],
       [ 1.39508604,  1.58316512,  1.36520822, ..., -1.52437837,
        -1.42894777, -0.59516041]])
```

In [9]:
```
pca = PCA()
pca_values = pca.fit_transform(df_norm)
pca_values
```

Out[9]:
```
array([[ 3.31675081e+00, -1.44346263e+00, -1.65739045e-01, ...,
        -4.51563395e-01,  5.40810414e-01, -6.62386309e-02],
       [ 2.20946492e+00,  3.33392887e-01, -2.02645737e+00, ...,
        -1.42657306e-01,  3.88237741e-01,  3.63650247e-03],
       [ 2.51674015e+00, -1.03115130e+00,  9.82818670e-01, ...,
        -2.86672847e-01,  5.83573183e-04,  2.17165104e-02],
       ...,
       [-2.67783946e+00, -2.76089913e+00, -9.40941877e-01, ...,
         5.12492025e-01,  6.98766451e-01,  7.20776948e-02],
       [-2.38701709e+00, -2.29734668e+00, -5.50696197e-01, ...,
         2.99821968e-01,  3.39820654e-01, -2.18657605e-02],
       [-3.20875816e+00, -2.76891957e+00,  1.01391366e+00, ...,
        -2.29964331e-01, -1.88787963e-01, -3.23964720e-01]])
```

In [10]:
```
var = pca.explained_variance_ratio_
var
```

Out[10]:
```
array([0.36198848, 0.1920749 , 0.11123631, 0.0706903 , 0.06563294,
       0.04935823, 0.04238679, 0.02680749, 0.02222153, 0.01930019,
       0.01736836, 0.01298233, 0.00795215])
```

In [11]:
```
var1 = np.cumsum(np.round(var,decimals = 4)*100)
var1
```

Out[11]:
```
array([ 36.2 ,  55.41,  66.53,  73.6 ,  80.16,  85.1 ,  89.34,  92
.02,
        94.24,  96.17,  97.91,  99.21, 100.01])
```

In [12]:
```
pca.components_
```

Out[12]:
```
array([[ 0.1443294 , -0.24518758, -0.00205106, -0.23932041,  0.141
99204,
         0.39466085,  0.4229343 , -0.2985331 ,  0.31342949, -0.088
6167 ,
         0.29671456,  0.37616741,  0.28675223],
       [-0.48365155, -0.22493093, -0.31606881,  0.0105905 , -0.299
634  ,
        -0.06503951,  0.00335981, -0.02877949, -0.03930172, -0.529
99567,
         0.27923515,  0.16449619, -0.36490283],
       [-0.20738262,  0.08901289,  0.6262239 ,  0.61208035,  0.130
75693,
         0.14617896,  0.1506819 ,  0.17036816,  0.14945431, -0.137
30621,
         0.08522192,  0.16600459, -0.12674592],
       [-0.0178563 ,  0.53689028, -0.21417556,  0.06085941, -0.351
79658,
         0.19806835,  0.15229479, -0.20330102,  0.39905653,  0.065
92568,
        -0.42777141,  0.18412074, -0.23207086],
       [-0.26566365,  0.03521363, -0.14302547,  0.06610294,  0.727
04851,
        -0.14931841, -0.10902584, -0.50070298,  0.13685982, -0.076
43678,
        -0.17361452, -0.10116099, -0.1578688 ],
       [-0.21353865, -0.53681385, -0.15447466,  0.10082451, -0.038
14394,
         0.0841223 ,  0.01892002,  0.25859401,  0.53379539,  0.418
64414,
        -0.10598274, -0.26585107, -0.11972557],
       [-0.05639636,  0.42052391, -0.14917061, -0.28696914,  0.322
8833 ,
        -0.02792498, -0.06068521,  0.59544729,  0.37213935, -0.227
71214,
         0.23207564, -0.0447637 ,  0.0768045 ],
       [-0.39613926, -0.06582674,  0.17026002, -0.42797018,  0.156
36143,
         0.40593409,  0.18724536,  0.23328465, -0.36822675,  0.033
79692,
        -0.43662362,  0.07810789, -0.12002267],
       [ 0.50861912, -0.07528304, -0.30769445,  0.20044931,  0.271
40257,
         0.28603452,  0.04957849,  0.19550132, -0.20914487,  0.056
21752,
         0.08582839,  0.1372269 , -0.57578611],
       [ 0.21160473, -0.30907994, -0.02712539,  0.05279942,  0.067
87022,
        -0.32013135, -0.16315051,  0.21553507,  0.1341839 , -0.290
77518,
        -0.52239889,  0.52370587,  0.162116  ],
       [-0.22591696,  0.07648554, -0.49869142,  0.47931378,  0.071
28891,
```

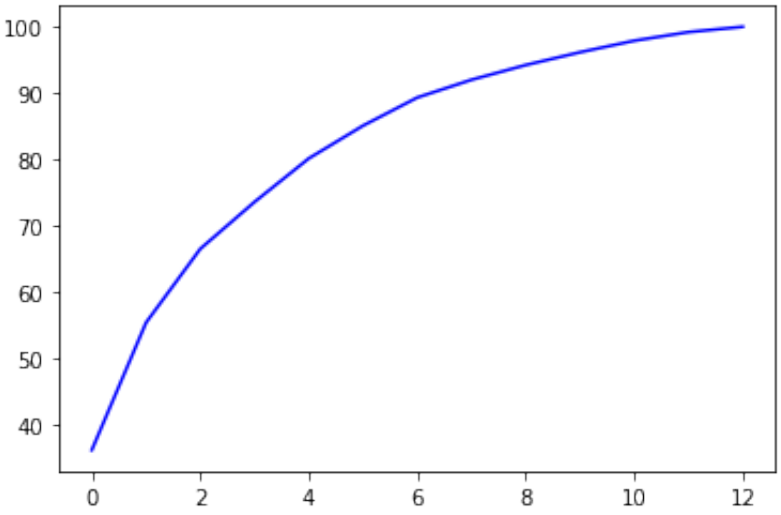
```
        0.30434119, -0.02569409,  0.11689586, -0.23736257,  0.031
8388 ,
       -0.04821201,  0.0464233 ,  0.53926983],
      [-0.26628645,  0.12169604, -0.04962237, -0.05574287,  0.062
22011,
       -0.30388245, -0.04289883,  0.04235219, -0.09555303,  0.604
22163,
        0.259214  ,  0.60095872, -0.07940162],
      [ 0.01496997,  0.02596375, -0.14121803,  0.09168285,  0.056
77422,
       -0.46390791,  0.83225706,  0.11403985, -0.11691707, -0.011
9928 ,
       -0.08988884, -0.15671813,  0.01444734]])
```

In [13]:
```
# variance
var=pca.explained_variance_ratio_
var
```

Out[13]:
```
array([0.36198848, 0.1920749 , 0.11123631, 0.0706903 , 0.06563294,
       0.04935823, 0.04238679, 0.02680749, 0.02222153, 0.01930019,
       0.01736836, 0.01298233, 0.00795215])
```

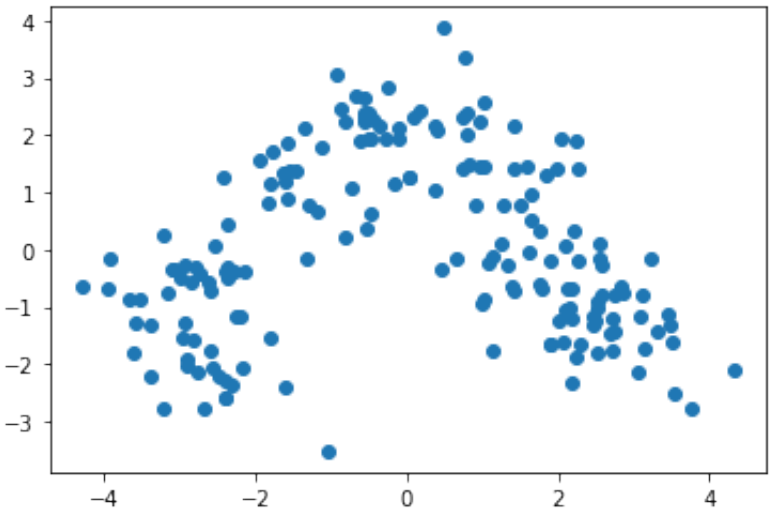
In [14]:
```
# variance plot for PCA components obtained
plt.plot(var1,color='blue')
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x7f9739d2d190>]
```



In [15]:
```
pca_values[:,0:2]
x = pca_values[:,0:1]
y = pca_values[:,1:2]
plt.scatter(x,y)
```

Out[15]: <matplotlib.collections.PathCollection at 0x7f9739c9b700>



In [16]:
```
pca_data = pd.DataFrame(pca_values[:,0:3],columns=["pca1","pca2","p
pca_data
```

Out[16]:

	pca1	pca2	pca3
0	3.316751	-1.443463	-0.165739
1	2.209465	0.333393	-2.026457
2	2.516740	-1.031151	0.982819
3	3.757066	-2.756372	-0.176192
4	1.008908	-0.869831	2.026688
...	...	...	...
173	-3.370524	-2.216289	-0.342570
174	-2.601956	-1.757229	0.207581
175	-2.677839	-2.760899	-0.940942
176	-2.387017	-2.297347	-0.550696
177	-3.208758	-2.768920	1.013914

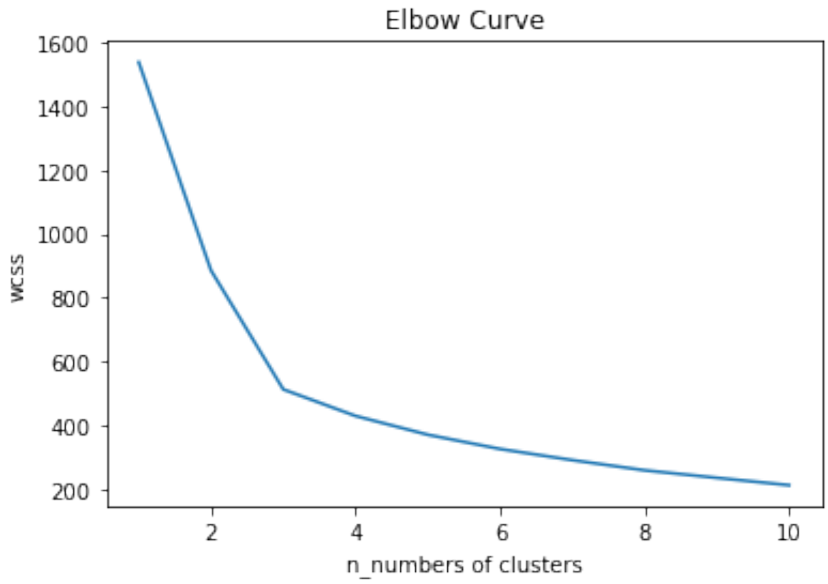
178 rows × 3 columns

In [17]:
```
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
```

ELBOW Curve

In [18]:
```
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(pca_data)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title("Elbow Curve")
plt.xlabel("n_numbers of clusters")
plt.ylabel("wcss")
```

Out[18]:
```
Text(0, 0.5, 'wcss')
```



Creating KMeans Model

In [19]:
```
model = KMeans(n_clusters=3)
model.fit(pca_data)
print(model.labels_)
km = pd.DataFrame(model.labels_,columns=["km_c"])
```

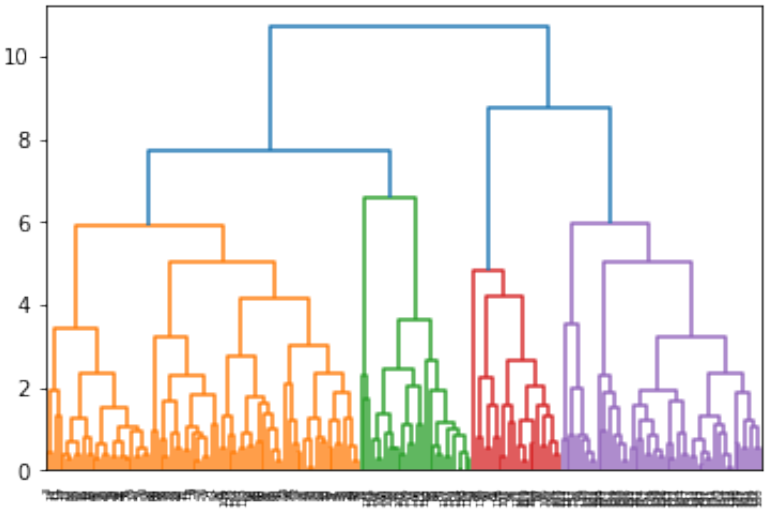
```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 0 1 1 1 1 1 1 1 1
1 1 1 2
 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1
1 1 1 1
 1 1 1 1 1 1 1 0 1 1 2 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

In [20]:
```
df_km = dataa.copy()
df_km["km_c"] = model.labels_
df_km.groupby(df_km.km_c).mean()
```

Out[20]:

km_c	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflav
0	13.134118	3.307255	2.417647	21.241176	98.666667	1.683922	0.818824	0.
1	12.249062	1.910313	2.233281	20.087500	92.812500	2.227813	2.023438	0.
2	13.656032	1.983175	2.460476	17.479365	107.650794	2.858254	3.015079	0.

In [21]:
```
dendrogram = sch.dendrogram(sch.linkage(pca_data,method="complete")
```



In [22]:
```
hc = AgglomerativeClustering(n_clusters = 3,affinity='euclidean',li
df_pred = hc.fit_predict(pca_data)
df_pred
hc = pd.DataFrame(df_pred,columns=["hc"])
hc.head()
```

Out[22]:

	hc
0	0
1	0
2	0
3	0
4	0

In [23]:
```
df_hc = dataa.copy()
df_hc['hc_Cluster'] = df_pred
df_hc.groupby(df_hc['hc_Cluster']).mean()
```

Out[23]:

hc_Cluster	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	N
0	13.065000	1.993396	2.406509	18.758491	101.990566	2.632075	2.663019	
1	13.115600	3.381800	2.449800	21.750000	98.480000	1.701000	0.844400	
2	12.429091	1.612727	1.984545	17.918182	91.772727	2.021818	1.668636	

In [24]:
```
final_clusters = pd.concat([data_cluster,hc,km],axis=1)
final_clusters
```

Out[24]:

	Type	hc	km_c
0	1	0	2
1	1	0	2
2	1	0	2
3	1	0	2
4	1	0	2
...	...	...	...
173	3	1	0
174	3	1	0
175	3	1	0
176	3	1	0
177	3	1	0

178 rows × 3 columns

In [ ]:
```

```