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

In [2]:
```
#data collection and reading
#decimal : set character used for decimal numbers
data = pd.read_csv("Project_Data_1.csv", index_col=0, decimal=',')
data.head()
```

Out[2]:

Sales of Wheat in tons	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002
Afghanistan	436.0	429.0	422.0	415.0	407.0	397.0	397	387	374	373	346	326	304
Albania	42.0	40.0	41.0	42.0	42.0	43.0	42	44	43	42	40	34	32
Algeria	45.0	44.0	44.0	43.0	43.0	42.0	43	44	45	46	48	49	50
American Samoa	42.0	14.0	4.0	18.0	17.0	22.0	0	25	12	8	8	6	5
Andorra	39.0	37.0	35.0	33.0	32.0	30.0	28	23	24	22	20	20	21

In [3]:
```
#check null, nan values
data.isna()
```

Out[3]:

Sales of Wheat in tons	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2(
Afghanistan	False	False	False	False	False	False	False	False	False	False	False	False	Fa
Albania	False	False	False	False	False	False	False	False	False	False	False	False	Fa
Algeria	False	False	False	False	False	False	False	False	False	False	False	False	Fa
American Samoa	False	False	False	False	False	False	False	False	False	False	False	False	Fa
Andorra	False	False	False	False	False	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	...	...	...	...	
Wallis et Futuna	False	False	False	False	False	False	False	False	False	False	False	False	Fa
West Bank and Gaza	False	False	False	False	False	False	False	False	False	False	False	False	Fa
Yemen	False	False	False	False	False	False	False	False	False	False	False	False	Fa
Zambia	False	False	False	False	False	False	False	False	False	False	False	False	Fa
Zimbabwe	False	False	False	False	False	False	False	False	False	False	False	False	Fa

207 rows × 18 columns

In [4]:
```
#drop nan values if exist
data.dropna()
```

Out[4]:

Sales of Wheat in tons	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002
Afghanistan	436.0	429.0	422.0	415.0	407.0	397.0	397	387	374	373	346	326	304
Albania	42.0	40.0	41.0	42.0	42.0	43.0	42	44	43	42	40	34	32
Algeria	45.0	44.0	44.0	43.0	43.0	42.0	43	44	45	46	48	49	50
American Samoa	42.0	14.0	4.0	18.0	17.0	22.0	0	25	12	8	8	6	5
Andorra	39.0	37.0	35.0	33.0	32.0	30.0	28	23	24	22	20	20	21
...	...	...	...	...	...	...	...	...	...	...	...	...	..
Wallis et Futuna	126.0	352.0	64.0	174.0	172.0	93.0	123	213	107	105	103	13	275
West Bank and Gaza	55.0	54.0	54.0	52.0	52.0	50.0	49	46	44	42	40	39	37
Yemen	265.0	261.0	263.0	253.0	250.0	244.0	233	207	194	175	164	154	149
Zambia	436.0	456.0	494.0	526.0	556.0	585.0	602	626	634	657	658	680	517
Zimbabwe	409.0	417.0	415.0	419.0	426.0	439.0	453	481	392	430	479	523	571

207 rows × 18 columns

In [5]:
```
#data splitting for clustering
data_for_decomposition = data.iloc[:, 0:]
data_for_decomposition.head()
```

Out[5]:

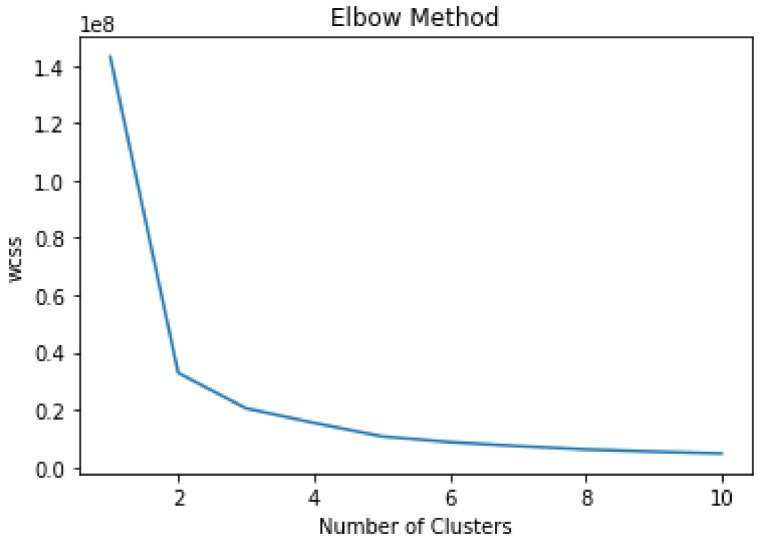
Sales of Wheat in tons	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002
Afghanistan	436.0	429.0	422.0	415.0	407.0	397.0	397	387	374	373	346	326	304
Albania	42.0	40.0	41.0	42.0	42.0	43.0	42	44	43	42	40	34	32
Algeria	45.0	44.0	44.0	43.0	43.0	42.0	43	44	45	46	48	49	50
American Samoa	42.0	14.0	4.0	18.0	17.0	22.0	0	25	12	8	8	6	5
Andorra	39.0	37.0	35.0	33.0	32.0	30.0	28	23	24	22	20	20	21

In [7]:
```
#PCA for dimension reduction
model_pca = PCA(n_components=2)
pca_data = model_pca.fit(data_for_decomposition).transform(data_for_decomposit
ion)

new_data = pd.DataFrame(pca_data, columns=["pca_1", "pca_2"])
new_data.index = data.index
new_data.head()
```

Out[7]:

Sales of Wheat in tons	pca_1	pca_2
Afghanistan	744.815213	-235.636419
Albania	-595.865592	6.105249
Algeria	-551.303760	45.952015
American Samoa	-700.700584	8.865202
Andorra	-645.423819	1.536970

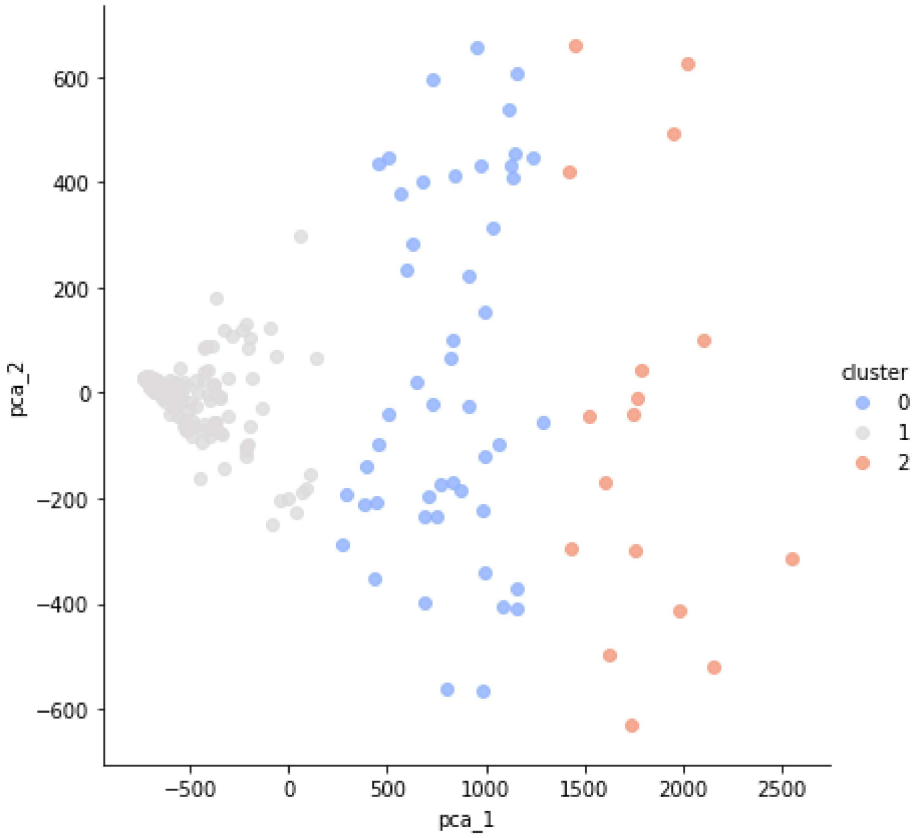
In [11]:
```
#elbow method to find number of clusters
wcss = []  #within cluster squared sum of inertia

for i in range(1,11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, r
andom_state=10)
    kmeans.fit(new_data)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11), wcss)
plt.title("Elbow Method")
plt.xlabel("Number of Clusters")
plt.ylabel("wcss")
plt.show()
#it indicates the number of clusters will be 3
```



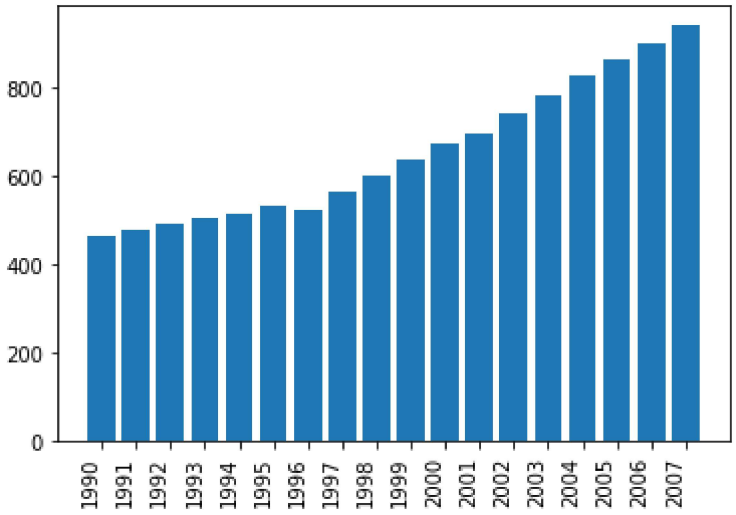
In [12]:
```
# Kmeans Algorithm
kmeans = KMeans(n_clusters=3)
# fir the model
kmeans.fit(new_data)
# clusters centers
print(kmeans.cluster_centers_)

# adding cluster column
new_data["cluster"] = kmeans.labels_
new_data.head()

# plotting the cluster data
sns.lmplot('pca_1', 'pca_2', data=new_data, hue='cluster',
           palette='coolwarm', height=6, aspect=1, fit_reg=False)
```

```
[[ 811.10727247   34.75475384]
 [-499.05863515   -5.74958477]
 [1801.3536003   -52.48773448]]
```

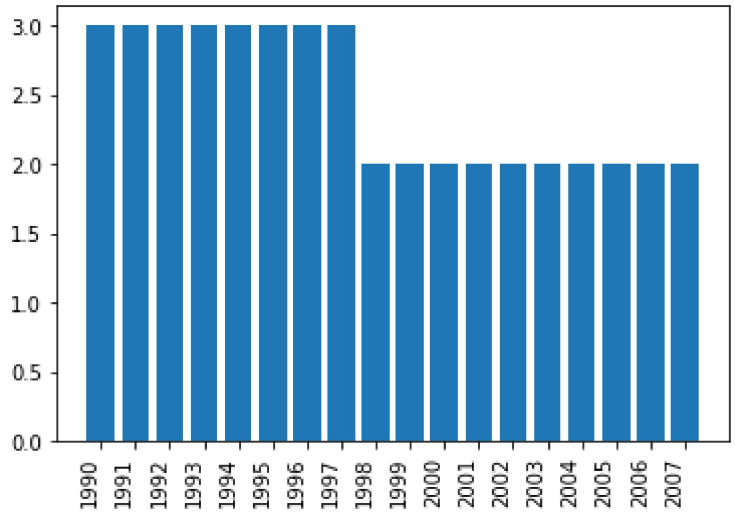
Out[12]: <seaborn.axisgrid.FacetGrid at 0x1cfb2ed7e88>



In [13]:
```
# adding cluster column to original data
data["cluster"] = kmeans.labels_


# export data for analysing
new_data.sort_values(["cluster", "pca_1", "pca_2"])
new_data.to_csv("output.csv", index=True)


# Largest Importer and constantly increasing
data.loc["Sierra Leone"]
X = data.loc["Sierra Leone"].index[0:18]
Y = data.loc["Sierra Leone"].values[0:18]

plt.bar(X, Y)
plt.setp(plt.gca().get_xticklabels(), rotation=90,
         horizontalalignment='right')  # Rotate Axis Labels

plt.show()
```



In [14]:
```
# most consistent
data.loc["Monaco"]
X = data.loc["Monaco"].index[0:18]
Y = data.loc["Monaco"].values[0:18]

plt.bar(X, Y)
plt.setp(plt.gca().get_xticklabels(), rotation=90,
         horizontalalignment='right')  # Rotate Axis Labels

plt.show()
```



In [ ]: