In [4]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [6]:
```python
data = pd.read_csv("Mall_Customers.csv")
data.head()
```

Out[6]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [8]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [10]:
```python
#CustomerID does not matter in segmentation , so lets drop the column
data = data.drop(columns = 'CustomerID')
data.head()
```

Out[10]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40

In [12]:
```python
#Lets consider male as 0 and female as 1 to get numeric value in Gender column
data = data.replace({'Gender' : {'Male':0 ,
                                 'Female':1}})
data.head()
```

Out[12]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	0	19	15	39
1	0	21	15	81
2	1	20	16	6
3	1	23	16	77
4	1	31	17	40

In [14]:
```python
data['Gender'].unique()
```

Out[14]:
```
array([0, 1], dtype=int64)
```

In [16]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Gender                  200 non-null    int64
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    int64
 3   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4)
memory usage: 6.4 KB
```

In [18]:
```python
#Normalize the data
from sklearn.preprocessing import Normalizer
norm = Normalizer()
columns = data.columns
norm_data = norm.fit_transform(data)

norm_data = pd.DataFrame(norm_data,columns = columns)
norm_data.head()
```

Out[18]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	0.000000	0.413925	0.326783	0.849635
1	0.000000	0.247025	0.176446	0.952809
2	0.037987	0.759737	0.607790	0.227921
3	0.012203	0.280676	0.195253	0.939653
4	0.018728	0.580581	0.318383	0.749137

In [20]:
```python
#Clustering Phase
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 2)
kmeans.fit(norm_data)
pred = kmeans.predict(norm_data)
```

In [22]:
```python
pred
```

Out[22]:
```
array([1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1])
```

In [24]:
```python
len(pred)
```

Out[24]:
```
200
```

In [26]:
```python
#Calculating the inertia
kmeans.inertia_
```

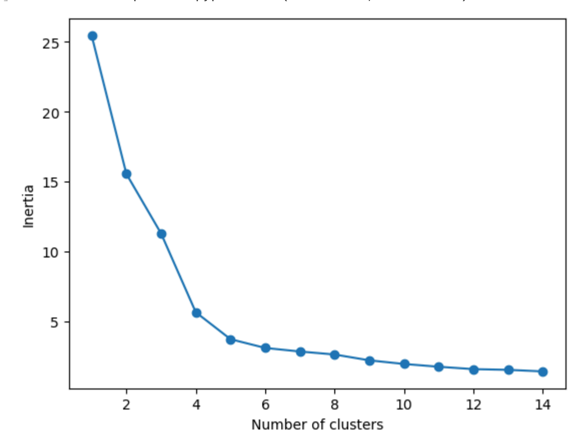
Out[26]:
```
15.554777496793392
```

In [28]:
```python
#To check the value of n_clusters
list = []
for cluster in range(1,15):
    kmeans = KMeans(n_clusters = cluster)
    kmeans.fit(norm_data)
    pred = kmeans.inertia_
    list.append(pred)
```

In [30]:
```python
frame = pd.DataFrame({'Cluster':range(1,15) , 'Inertia':list})
```

In [32]:
```python
plt.plot(figsize=(7,5))
plt.plot(frame['Cluster'],frame['Inertia'],marker='o')
plt.xlabel("Number of clusters")
plt.ylabel("Inertia")
plt.show
```

Out[32]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [33]:
```python
#We take number of clusters as 5
kmeans = KMeans(n_clusters = 5)
kmeans.fit(norm_data)
predictions = kmeans.predict(norm_data)
```

In [36]:
```python
norm_data['Cluster'] = predictions
print(norm_data)
```

```
       Gender       Age  Annual Income (k$)  Spending Score (1-100)  Cluster
0    0.000000  0.413925            0.326783                0.849635        3
1    0.000000  0.247025            0.176446                0.952809        3
2    0.037987  0.759737            0.607790                0.227921        4
3    0.012203  0.280676            0.195253                0.939653        3
4    0.018728  0.580581            0.318383                0.749137        3
..        ...       ...                 ...                     ...      ...
195  0.006762  0.236686            0.811496                0.534235        1
196  0.007315  0.329196            0.921748                0.204833        2
197  0.000000  0.213924            0.842324                0.494698        1
198  0.000000  0.225615            0.965915                0.126909        2
199  0.000000  0.184087            0.840665                0.509308        1

[200 rows x 5 columns]
```

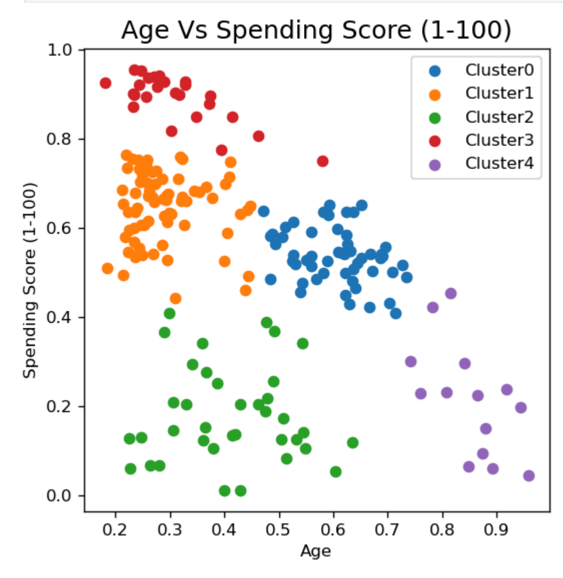
In [38]:
```python
#Plot 2 features against each other at a time
def seg(str_x,str_y,clusters):
    x=[]
    y=[]

    for i in range(clusters):
        x.append(norm_data[str_x][norm_data['Cluster'] == i])
        y.append(norm_data[str_y][norm_data['Cluster'] == i])

    return x,y

def plot_clusters(str_x,str_y,clusters):
    plt.figure(figsize=(5,5),dpi=120)
    x,y = seg(str_x,str_y,clusters)

    for i in range(clusters):
        plt.scatter(x[i],y[i],label='Cluster{}'.format(i))

    plt.xlabel(str_x,fontsize = 10)
    plt.ylabel(str_y,fontsize = 10)
    plt.title(str(str_x+" Vs "+str_y),fontsize = 15)
    plt.legend()
```

In [40]:
```python
norm_data.head()
```

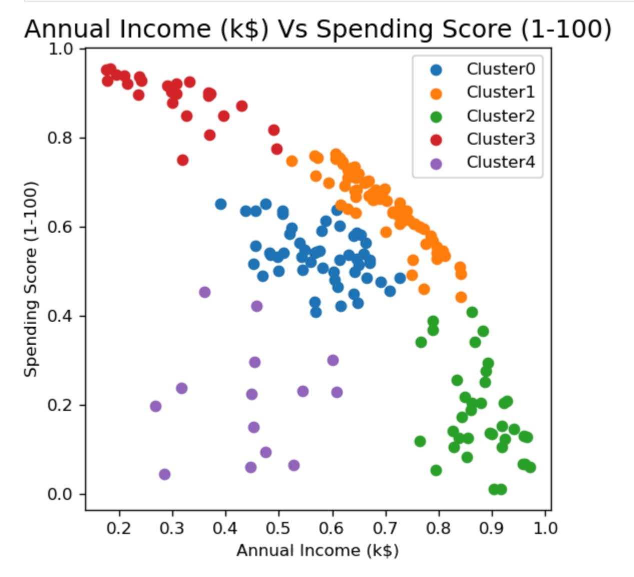
Out[40]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	0.000000	0.413925	0.326783	0.849635	3
1	0.000000	0.247025	0.176446	0.952809	3
2	0.037987	0.759737	0.607790	0.227921	4
3	0.012203	0.280676	0.195253	0.939653	3
4	0.018728	0.580581	0.318383	0.749137	3

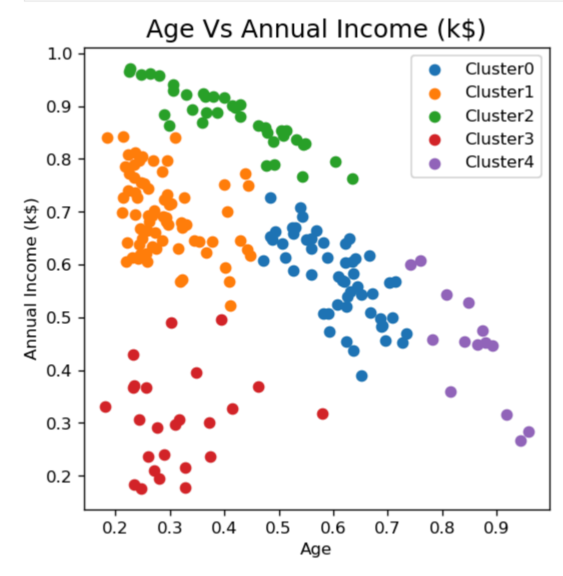
In [42]:
```python
#Age vs Spending Score
plot_clusters('Age','Spending Score (1-100)',5)
```



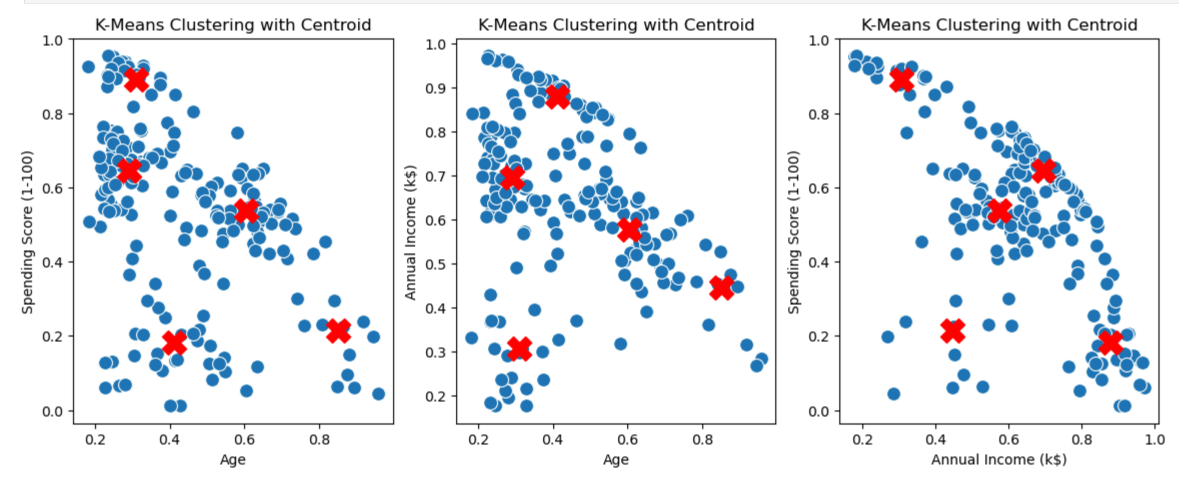
In [43]:
```python
#Annual Income vs Spending Score
plot_clusters('Annual Income (k$)','Spending Score (1-100)',5)
```



In [45]:
```python
#Age vs Annual Income
plot_clusters('Age','Annual Income (k$)',5)
```



In [50]:
```python
# Plotting the results
feature_x = 'Age'
feature_y = 'Annual Income (k$)'
feature_z = 'Spending Score (1-100)'

plt.figure(figsize=(14,5))
plt.subplot(1,3,1)
sns.scatterplot(data=norm_data,x = feature_x, y=feature_z ,palette='viridis', s=100)
plt.scatter(kmeans.cluster_centers_[:, 1], kmeans.cluster_centers_[:, 3], s=300, c='red', marker='X')
plt.title('K-Means Clustering with Centroid')
plt.xlabel(feature_x)
plt.ylabel(feature_z)

plt.subplot(1,3,2)
sns.scatterplot(data=norm_data,x = feature_x, y=feature_y ,palette='viridis', s=100)
plt.scatter(kmeans.cluster_centers_[:, 1], kmeans.cluster_centers_[:, 2], s=300, c='red', marker='X')
plt.title('K-Means Clustering with Centroid')
plt.xlabel(feature_x)
plt.ylabel(feature_y)

plt.subplot(1,3,3)
sns.scatterplot(data=norm_data,x = feature_y, y=feature_z ,palette='viridis', s=100)
plt.scatter(kmeans.cluster_centers_[:, 2], kmeans.cluster_centers_[:, 3], s=300, c='red', marker='X')
plt.title('K-Means Clustering with Centroid')
plt.xlabel(feature_y)
plt.ylabel(feature_z)
plt.show()
```



In [66]:
```python
kmeans.labels_
```

Out[66]:
```
array([3, 3, 4, 3, 3, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 0, 3, 4, 3, 0, 3,
       4, 3, 4, 3, 0, 3, 0, 3, 4, 3, 4, 3, 4, 3, 4, 3, 0, 3, 4, 3, 0, 3,
       0, 3, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1,
       2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1,
       2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1,
       2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1,
       2, 1])
```

In [60]:
```python
data['Label'] = kmeans.labels_
data.head()
```

Out[60]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Label
0	0	19	15	39	3
1	0	21	15	81	3
2	1	20	16	6	4
3	1	23	16	77	3
4	1	31	17	40	3

In [62]:
```python
data.Label.value_counts()
```

Out[62]:
```
Label
1    71
0    53
2    37
3    25
4    14
Name: count, dtype: int64
```

In [64]:
```python
for k in range(5):
    print(f'cluster nb : {k}')
    print(data[data['Label'] == k].describe())
    print('\n\n')
```

```
cluster nb : 0
          Gender        Age  Annual Income (k$)  Spending Score (1-100)  Label
count  53.000000  53.000000           53.000000               53.000000   53.0
mean    0.584906  53.075472           50.867925               47.000000    0.0
std     0.497454  10.306550           11.101001                8.107451    0.0
min     0.000000  31.000000           21.000000               26.000000    0.0
25%     0.000000  47.000000           44.000000               42.000000    0.0
50%     1.000000  50.000000           54.000000               48.000000    0.0
75%     1.000000  63.000000           60.000000               52.000000    0.0
max     1.000000  70.000000           67.000000               60.000000    0.0



cluster nb : 1
          Gender        Age  Annual Income (k$)  Spending Score (1-100)  Label
count  71.000000  71.000000           71.000000               71.000000   71.0
mean    0.577465  29.732394           73.140845               67.830986    1.0
std     0.497479   6.212793           20.090009               17.864559    0.0
min     0.000000  18.000000           40.000000               40.000000    1.0
25%     0.000000  27.000000           60.000000               53.000000    1.0
50%     1.000000  30.000000           71.000000               69.000000    1.0
75%     1.000000  34.000000           83.000000               85.500000    1.0
max     1.000000  40.000000          137.000000               97.000000    1.0



cluster nb : 2
          Gender        Age  Annual Income (k$)  Spending Score (1-100)  Label
count  37.000000  37.000000           37.000000               37.000000   37.0
mean    0.459459  40.729730           87.297297               18.054054    2.0
std     0.505228  11.342273           16.390825               10.461426    0.0
min     0.000000  19.000000           70.000000                1.000000    2.0
25%     0.000000  34.000000           77.000000               10.000000    2.0
50%     0.000000  42.000000           81.000000               16.000000    2.0
75%     1.000000  47.000000           97.000000               26.000000    2.0
max     1.000000  59.000000          137.000000               39.000000    2.0



cluster nb : 3
       Gender        Age  Annual Income (k$)  Spending Score (1-100)  Label
count    25.0  25.000000           25.000000                25.00000   25.0
mean      0.6  25.480000           25.480000                75.44000    3.0
std       0.5   5.339476            8.083522                15.06674    0.0
min       0.0  18.000000           15.000000                39.00000    3.0
25%       0.0  21.000000           19.000000                72.00000    3.0
50%       1.0  24.000000           24.000000                76.00000    3.0
75%       1.0  30.000000           33.000000                82.00000    3.0
max       1.0  35.000000           39.000000                99.00000    3.0



cluster nb : 4
          Gender        Age  Annual Income (k$)  Spending Score (1-100)  Label
count  14.000000  14.000000           14.000000               14.000000   14.0
mean    0.571429  50.142857           25.428571               12.785714    4.0
std     0.513553  13.236646            7.154496                9.552918    0.0
min     0.000000  20.000000           16.000000                3.000000    4.0
25%     0.000000  43.000000           19.250000                5.250000    4.0
50%     1.000000  52.500000           24.000000               13.500000    4.0
75%     1.000000  59.500000           32.250000               14.750000    4.0
max     1.000000  67.000000           38.000000               35.000000    4.0
```

In [ ]: