In [49]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
```

In [50]:

```
df= pd.read_csv(r"C:\Users\dtdee\OneDrive\Desktop\Letsupgrade_Python\Machine_Learning\KMeans\Mall_Customers_Unsupervised.csv
```

In [51]:

```
df.rename(columns={'Annual_Income_(k$)':'Income'},inplace=True)
```

In [52]:

```
df.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   Genre           200 non-null    object
 2   Age             200 non-null    int64 
 3   Income          200 non-null    int64 
 4   Spending_Score  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [53]:

```
df.isnull().sum()
```

Out[53]:

```
CustomerID        0
Genre             0
Age               0
Income            0
Spending_Score    0
dtype: int64
```

In [54]:

```
df.duplicated().sum()
```

Out[54]:

```
0
```

In [55]:

```
df.describe().T
```

Out[55]:

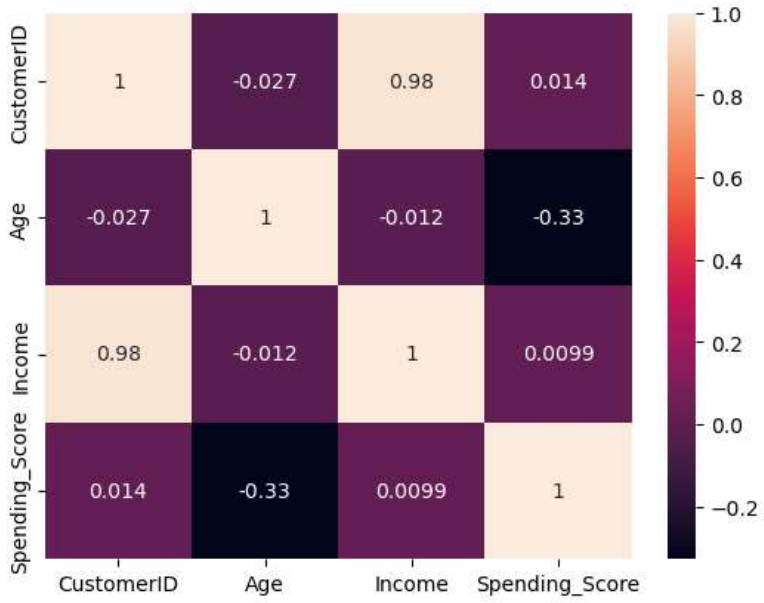
	count	mean	std	min	25%	50%	75%	max
CustomerID	200.0	100.50	57.879185	1.0	50.75	100.5	150.25	200.0
Age	200.0	38.85	13.969007	18.0	28.75	36.0	49.00	70.0
Income	200.0	60.56	26.264721	15.0	41.50	61.5	78.00	137.0
Spending_Score	200.0	50.20	25.823522	1.0	34.75	50.0	73.00	99.0

In [56]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[56]:

<AxesSubplot:>



In [57]:

```
# We see that Income and Spending _Score are the two most important columns and have very strong coorealtion with each other

df=df[['Income','Spending_Score']]
df
```

Out[57]:

	Income	Spending_Score
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40
...	...	...
195	120	79
196	126	28
197	126	74
198	137	18
199	137	83

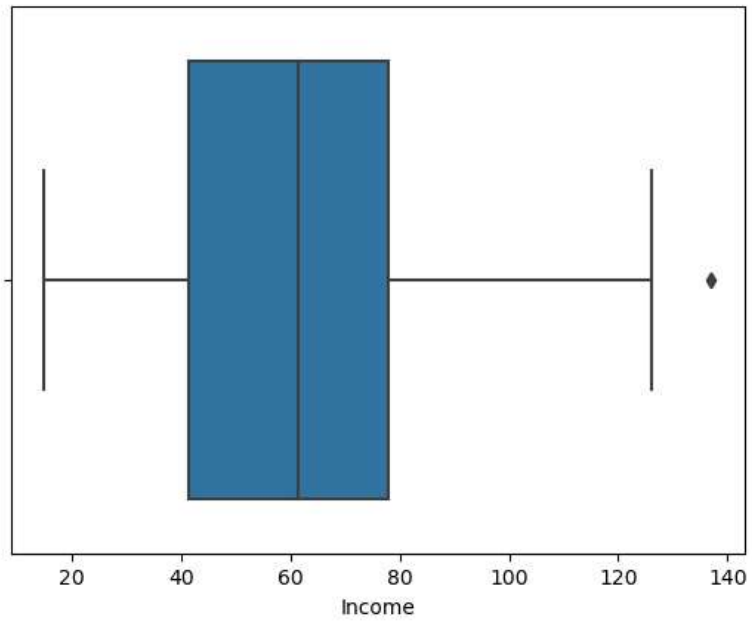
200 rows × 2 columns

In [58]:

```
sns.boxplot(x='Income',data=df)
```

Out[58]:

```
<AxesSubplot:xlabel='Income'>
```

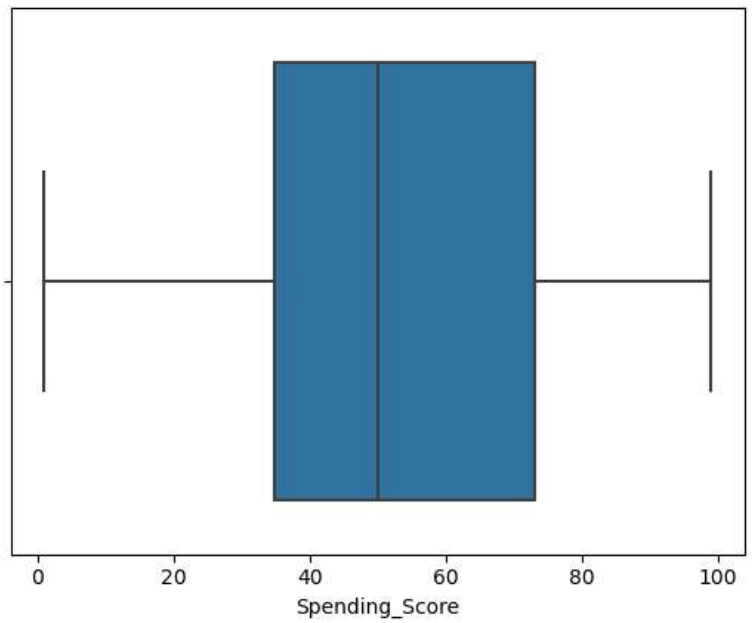


In [59]:

```
sns.boxplot(x='Spending_Score',data=df)
```

Out[59]:

```
<AxesSubplot:xlabel='Spending_Score'>
```



Implementing both Unpervised Algorithms one by one

BY KMEANS MODEL

In [ ]:

```

```

In [62]:

```
loss=[]

for i in range (1,10):
    kmeans=KMeans(n_clusters=i,max_iter=100,random_state=40,init='k-means++')
    kmeans.fit(df)
    loss.append(kmeans.inertia_)
    print('Loss of the model is=',loss)
```

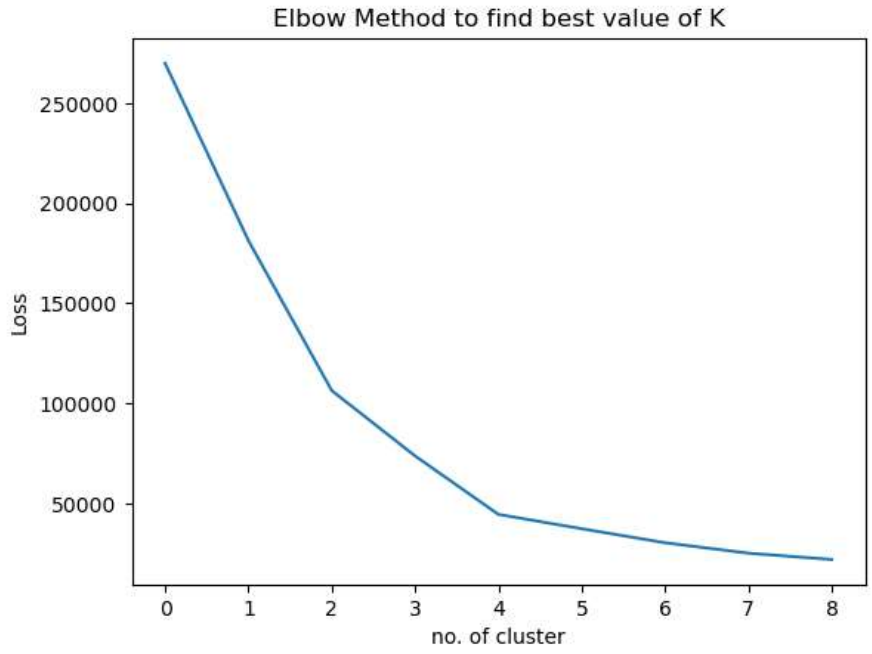
```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to hav
e a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by se
tting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(

Loss of the model is= [269981.28000000014]
Loss of the model is= [269981.28000000014, 181363.59595959607]
Loss of the model is= [269981.28000000014, 181363.59595959607, 106348.37306211119]
Loss of the model is= [269981.28000000014, 181363.59595959607, 106348.37306211119, 73679.78903948837]
Loss of the model is= [269981.28000000014, 181363.59595959607, 106348.37306211119, 73679.78903948837, 44448.45
544793369]
Loss of the model is= [269981.28000000014, 181363.59595959607, 106348.37306211119, 73679.78903948837, 44448.45
544793369, 37265.86520484345]
Loss of the model is= [269981.28000000014, 181363.59595959607, 106348.37306211119, 73679.78903948837, 44448.45
544793369, 37265.86520484345, 30259.657207285458]
Loss of the model is= [269981.28000000014, 181363.59595959607, 106348.37306211119, 73679.78903948837, 44448.45
544793369, 37265.86520484345, 30259.657207285458, 25044.96776401891]
Loss of the model is= [269981.28000000014, 181363.59595959607, 106348.37306211119, 73679.78903948837, 44448.45
544793369, 37265.86520484345, 30259.657207285458, 25044.96776401891, 21884.744095710266]
```

In [64]:

```
plt.plot(loss)
plt.xlabel('no. of cluster')
plt.ylabel('Loss')
plt.title('Elbow Method to find best value of K')
```

Out[64]:

```
Text(0.5, 1.0, 'Elbow Method to find best value of K')
```



In [65]:

```
# Now,Let us take the value of k as 4

model=KMeans(n_clusters=4,random_state=40,init='k-means++')
model.fit(df)
```

Out[65]:

```
KMeans(n_clusters=4, random_state=40)
```

In [67]:

```
model.inertia_
```

Out[67]:

```
73679.78903948837
```

In [70]:

```
# Classes or clusters generated after model is trained on the dataset are below

model.labels_
```

Out[70]:

```
array([3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0,
       3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0,
       3, 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, 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, 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, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
       1, 2])
```

In [69]:

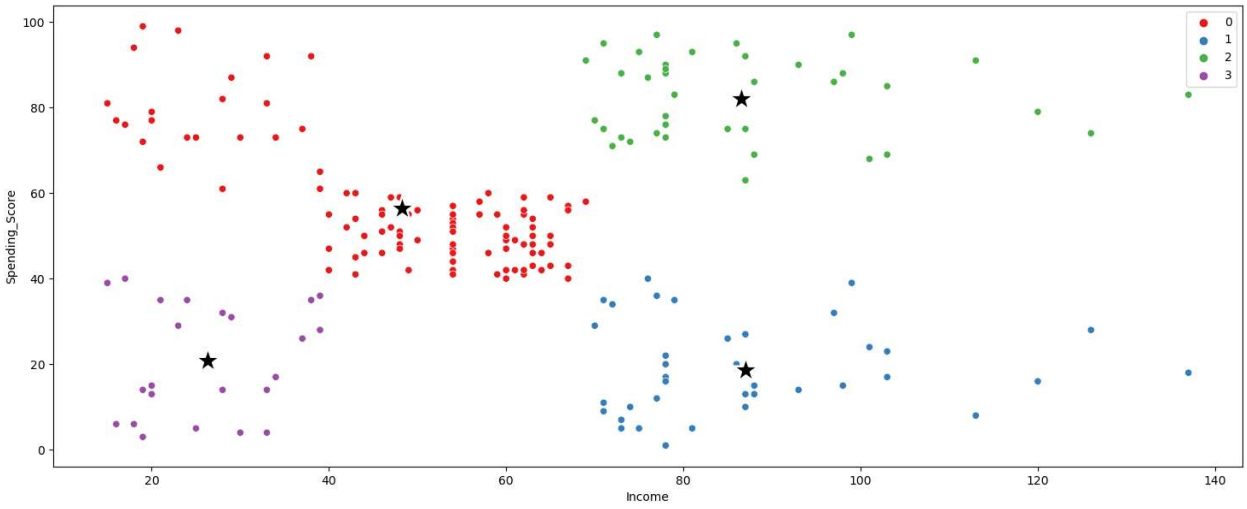
```
cen=model.cluster_centers_
```

In [80]:

```
# Plotting the clusters with their centriod values

plt.subplots(figsize=(18,7))
sns.scatterplot(x='Income'  ,y= 'Spending_Score',data=df,hue=model.labels_ ,palette='Set1')
sns.scatterplot(x= cen[:,0] ,y= cen[:,1] ,marker='*',s=500,color='Black')
```

Out[80]:

```
<AxesSubplot:xlabel='Income', ylabel='Spending_Score'>
```



BY DBSCAN METHOD

In [93]:

```
model1= DBSCAN(eps= 2 ,min_samples= 8 )
model1.fit(df)
model1.labels_
```

Out[93]:

```
array([-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, -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, -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, -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, -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, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1], dtype=int64)
```

In [94]:

```
model2= DBSCAN(eps= 4 ,min_samples= 8 )
model2.fit(df)
model2.labels_
```

Out[94]:

```
array([-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, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1,  0, -1, -1, -1,  1, -1,  1, -1,  0,  0,  1,  0,  1,  0,  1,  1,
        0,  1,  0, -1,  1,  0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,
       -1,  2, -1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2, -1,  2,
       -1, -1,  2, -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, -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, -1, -1, -1, -1], dtype=int64)
```

In [95]:

```
model3= DBSCAN(eps= 5 ,min_samples= 8 )
model3.fit(df)
model3.labels_
```

Out[95]:

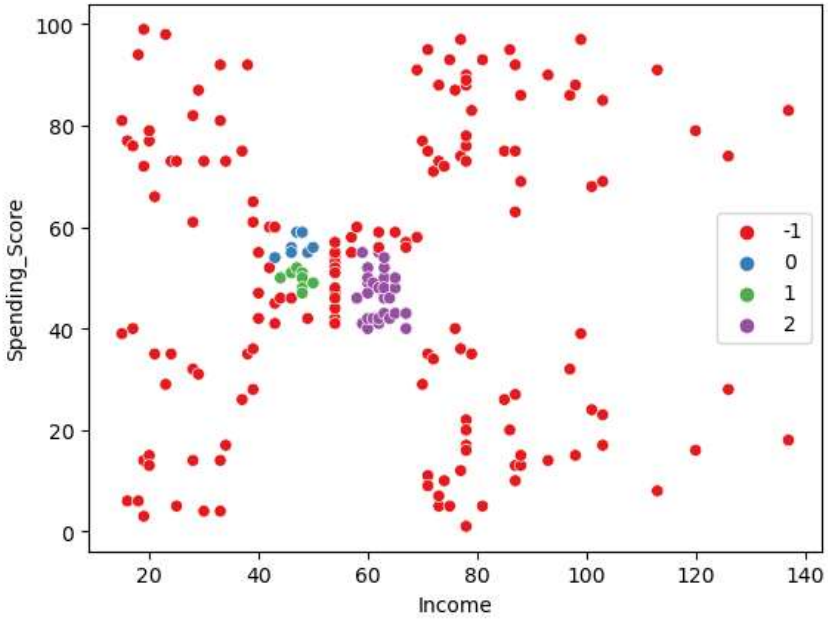
```
array([-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  0,  0, -1, -1,  0,
       -1,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  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,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  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,  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, -1, -1, -1, -1], dtype=int64)
```

In [104]:

```
# So now lets take the best cluster formed is 4 so we will try to plot this

sns.scatterplot(x='Income',y='Spending_Score',data=df,hue=model2.labels_ ,palette='Set1')
```

Out[104]:

```
<AxesSubplot:xlabel='Income', ylabel='Spending_Score'>
```

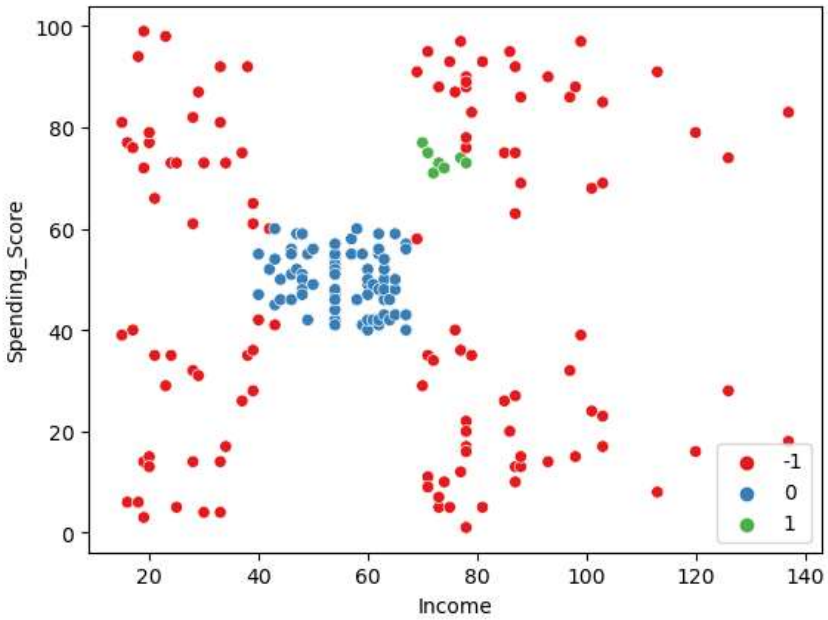


In [106]:

```
# So now lets take the best cluster formed is 3 so we will try to plot this

sns.scatterplot(x='Income',y='Spending_Score',data=df,hue=model3.labels_ ,palette='Set1')
```

Out[106]:

```
<AxesSubplot:xlabel='Income', ylabel='Spending_Score'>
```



In [ ]: