Vignesh M - TM1 (Assignment - 4)

Date : 01/11/2022

Team ID: PNT2022TMID38676

In [44]:
```
import numpy as np
import seaborn as sns
sns.set()
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.metrics import classification_report,accuracy_score,f1_score,hamming_
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

1.Download the dataset

2.Load the dataset into the tool

In [45]:
```
df = pd.read_csv("Mall_Customers.csv")
df.head(10)
```

Out[45]:

	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
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

In [46]:
```
df.drop(['CustomerID'],axis=1,inplace=True)
df.head(10)
```

Out[46]:

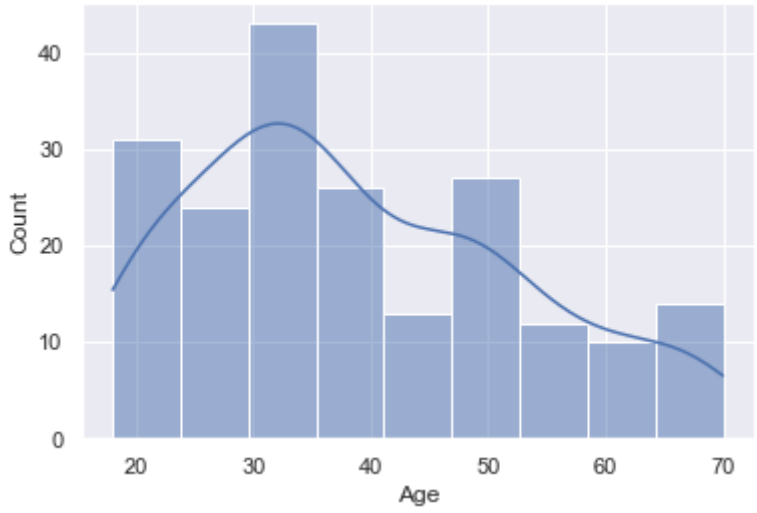
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40
5	Female	22	17	76
6	Female	35	18	6
7	Female	23	18	94
8	Male	64	19	3
9	Female	30	19	72

3.Perform below visualizations

3.1) Univariate Analysis

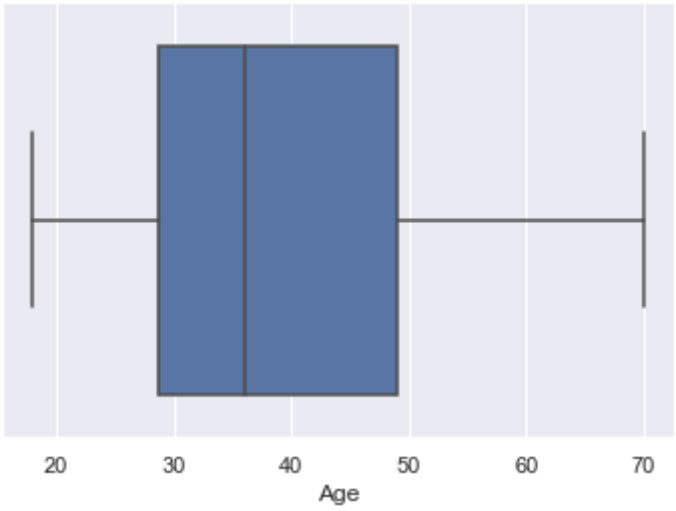
In [47]:
```
sns.histplot(df['Age'],kde=True)
```

Out[47]:
```
<AxesSubplot:xlabel='Age', ylabel='Count'>
```



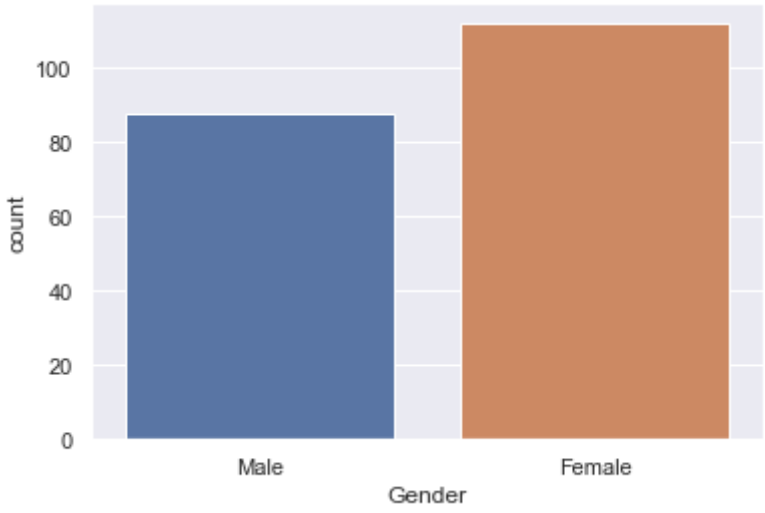
In [48]:
```
sns.boxplot(df['Age'],orient='w')
```

Out[48]:
```
<AxesSubplot:xlabel='Age'>
```



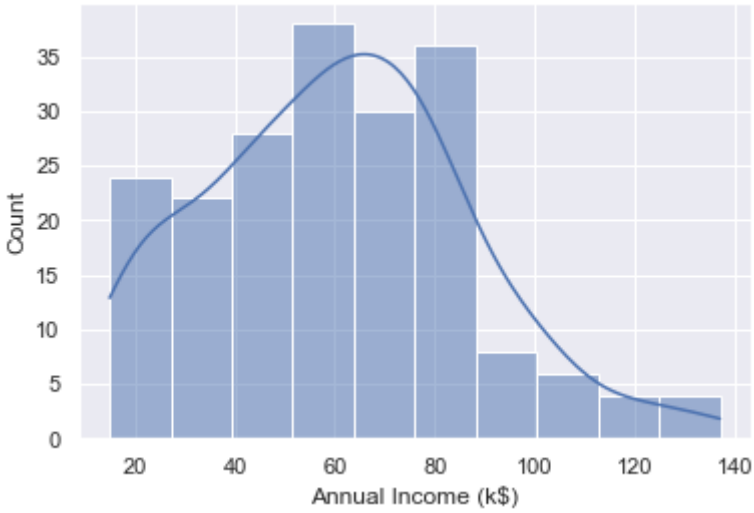
In [49]:
```
sns.countplot(x='Gender',data=df)
```

Out[49]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



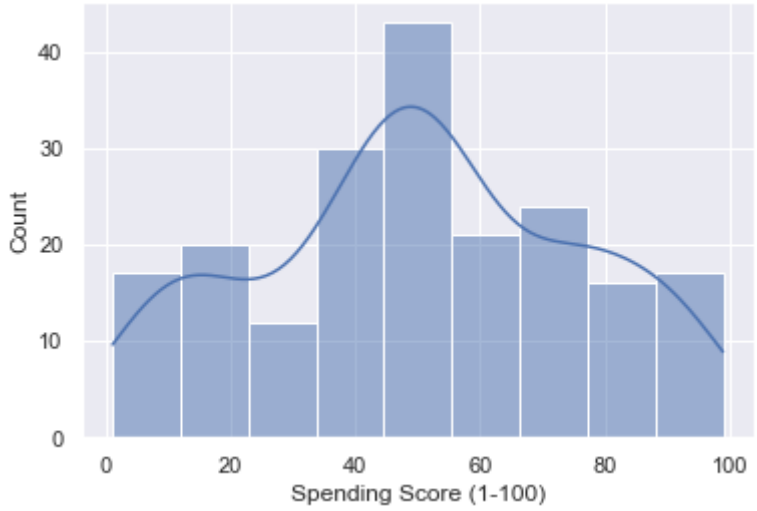
In [50]:
```
sns.histplot(df['Annual Income (k$)'], kde=True)
```

Out[50]:
```
<AxesSubplot:xlabel='Annual Income (k$)', ylabel='Count'>
```



In [51]:
```
sns.histplot(df['Spending Score (1-100)'], kde=True)
```

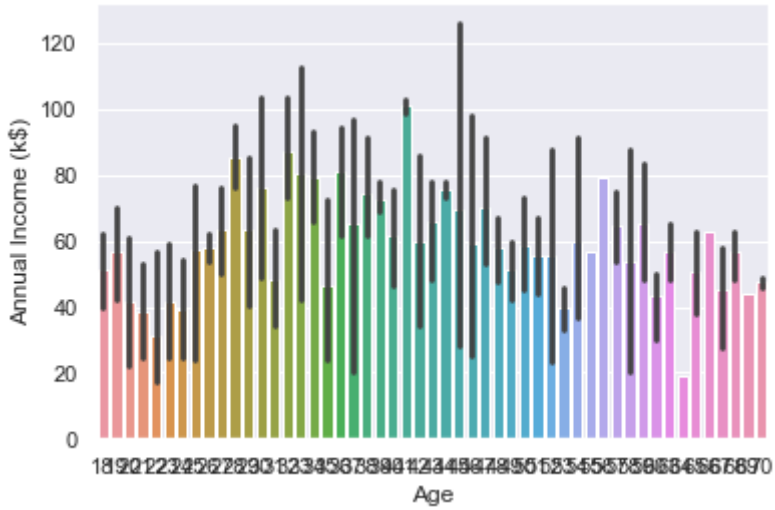
Out[51]:
```
<AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Count'>
```



3.2) Bi- Variate Analysis

In [52]:
```
sns.barplot(x='Age',y='Annual Income (k$)',data=df)
```

Out[52]:
```
<AxesSubplot:xlabel='Age', ylabel='Annual Income (k$)'>
```

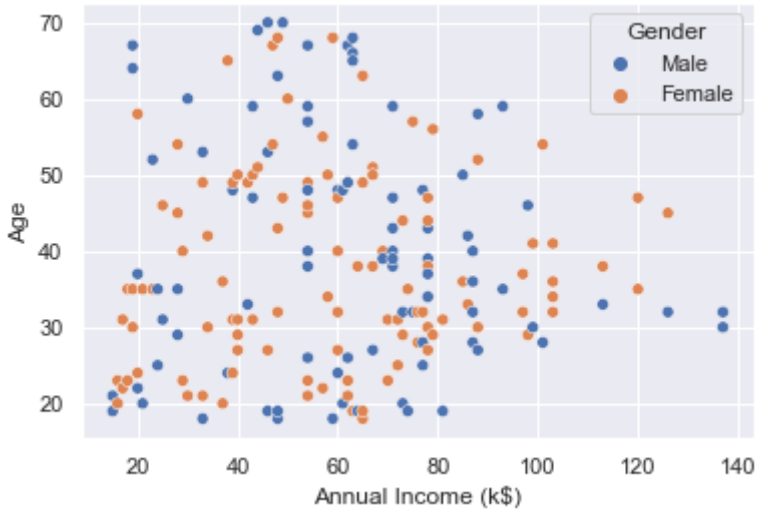


In [53]:
```
sns.lineplot(x='Spending Score (1-100)', y='Annual Income (k$)', data=df)
```

Out[53]:
```
<AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Annual Income (k$)'>
```

In [54]:
```
sns.scatterplot(x='Annual Income (k$)',y='Age',hue='Gender',data=df)
```

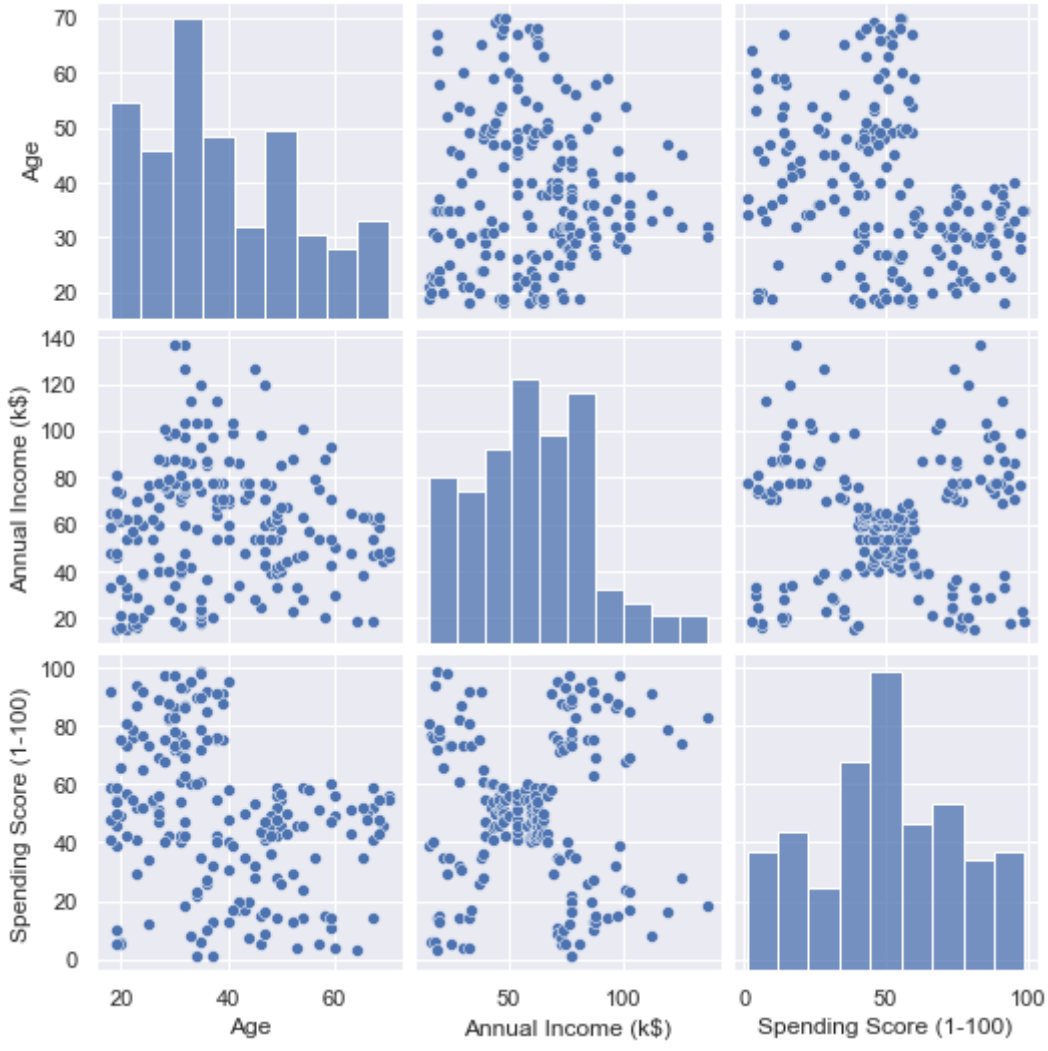
Out[54]:
```
<AxesSubplot:xlabel='Annual Income (k$)', ylabel='Age'>
```



3.3) Multi-Variate Analysis

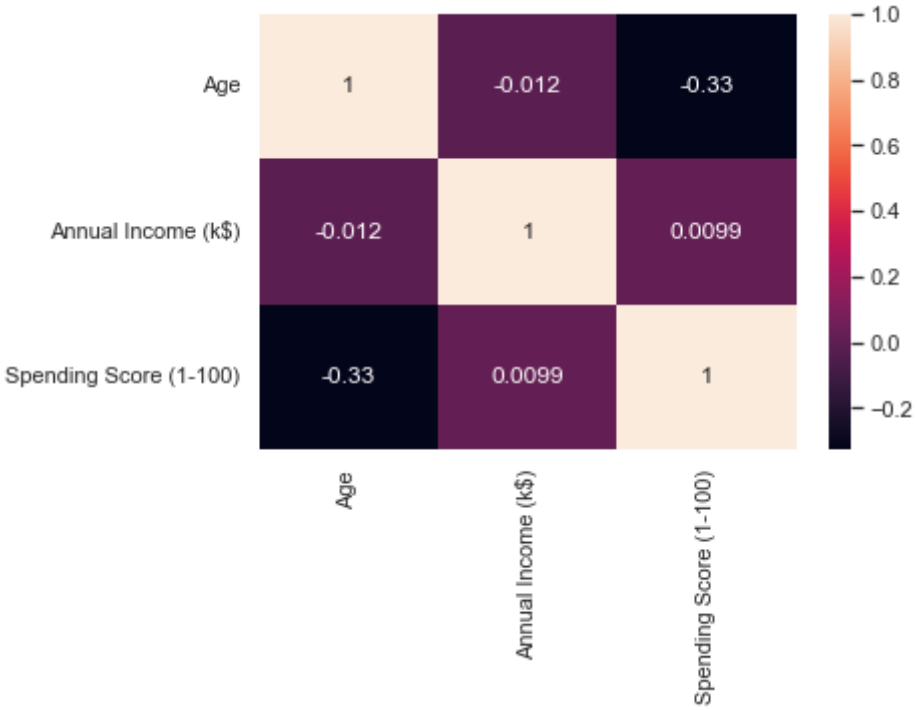
In [55]:
```
sns.pairplot(data=df[["Gender", "Age","Annual Income (k$)","Spending Score (1-100)"
```

Out[55]:
```
<seaborn.axisgrid.PairGrid at 0x22115650a90>
```



In [56]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[56]: <AxesSubplot:>



4.Perform descriptive statistics on the dataset.

In [57]:
```
df.describe()
```

Out[57]:

	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000
mean	38.850000	60.560000	50.200000
std	13.969007	26.264721	25.823522
min	18.000000	15.000000	1.000000
25%	28.750000	41.500000	34.750000
50%	36.000000	61.500000	50.000000
75%	49.000000	78.000000	73.000000
max	70.000000	137.000000	99.000000

5.Check for Missing values and deal with them

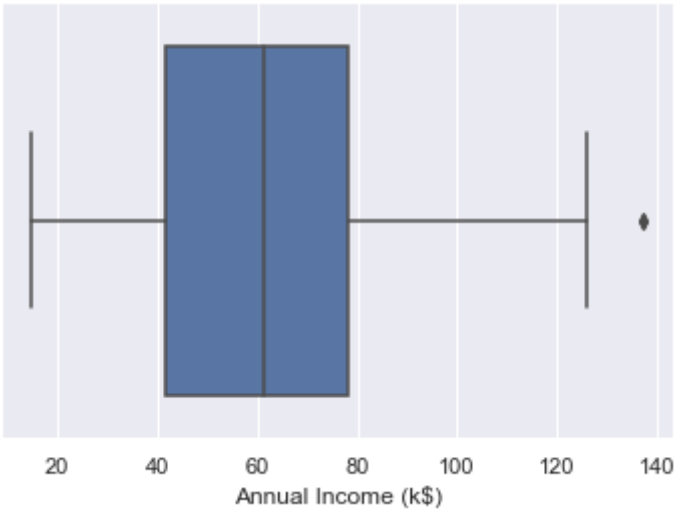
In [58]:
```
df.isnull().sum()
```

Out[58]:
```
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

6.Find the outliers and replace them outliers

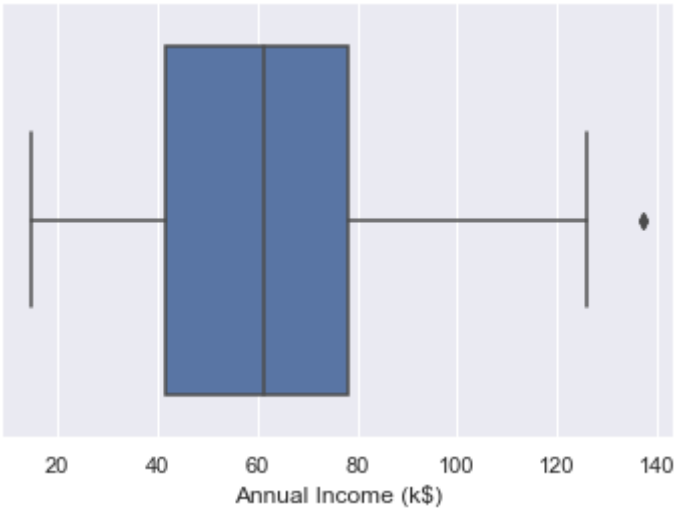
In [59]:
```
sns.boxplot(df['Annual Income (k$)'], orient='h')
```

Out[59]:
```
<AxesSubplot:xlabel='Annual Income (k$)'>
```



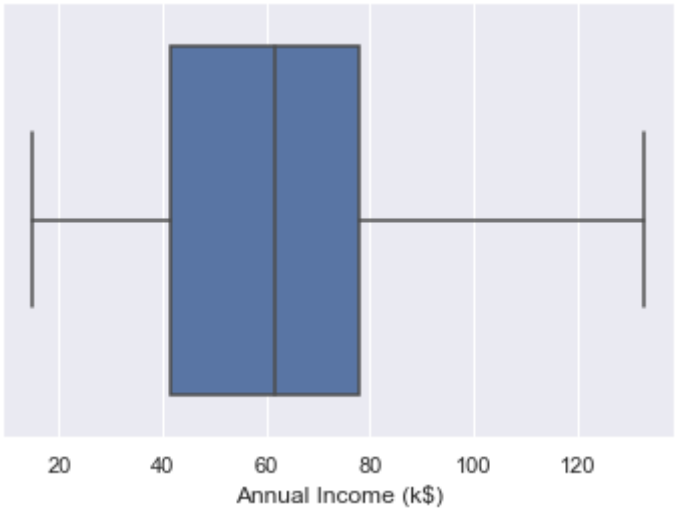
In [60]:
```
sns.boxplot(df['Annual Income (k$)'], orient='h')
```

Out[60]:
```
<AxesSubplot:xlabel='Annual Income (k$)'>
```



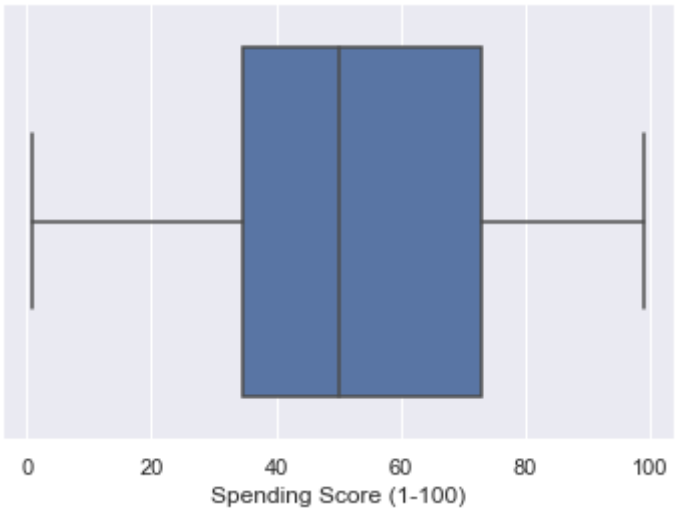
In [61]:
```
q = df['Annual Income (k$)'].quantile(q=[0.75,0.25])
iqr=q.iloc[0]-q.iloc[1]
lower = q.iloc[1] - 1.5*iqr
upper = q.iloc[0] + 1.5*iqr
df['Annual Income (k$)'] = np.where(df['Annual Income (k$)']>upper,upper,np.where(
sns.boxplot(df['Annual Income (k$)'], orient='h')
```

Out[61]:
```
<AxesSubplot:xlabel='Annual Income (k$)'>
```



In [62]:
```
sns.boxplot(df['Spending Score (1-100)'], orient='h')
```

Out[62]:
```
<AxesSubplot:xlabel='Spending Score (1-100)'>
```



7.Check for Categorical columns and perform encoding.

In [63]:
```
l_en = LabelEncoder()
df['Gender'] = l_en.fit_transform(df['Gender'])
df.head(10)
```

Out[63]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15.0	39
1	1	21	15.0	81
2	0	20	16.0	6
3	0	23	16.0	77
4	0	31	17.0	40
5	0	22	17.0	76
6	0	35	18.0	6
7	0	23	18.0	94
8	1	64	19.0	3
9	0	30	19.0	72

8.Scaling the data

In [64]:
```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
scaled_data[1:1]
```

Out[64]:
```
array([], shape=(0, 4), dtype=float64)
```

9.Perform any of the clustering algorithms

In [65]:
```
from sklearn.cluster import KMeans
km = KMeans(algorithm='elkan',n_init=100, max_iter=3000)
res = km.fit_predict(scaled_data)
res
```

Out[65]:
```
array([3, 3, 7, 7, 7, 7, 6, 7, 2, 7, 2, 7, 6, 7, 5, 3, 7, 3, 2, 7, 3, 3,
       6, 3, 6, 3, 6, 3, 6, 7, 2, 7, 2, 3, 6, 7, 6, 7, 6, 7, 6, 3, 2, 7,
       6, 7, 6, 7, 7, 7, 6, 3, 7, 2, 6, 2, 6, 2, 7, 2, 2, 3, 6, 6, 2, 3,
       6, 6, 3, 7, 2, 6, 6, 6, 2, 3, 6, 3, 7, 6, 2, 3, 2, 6, 7, 2, 6, 7,
       7, 6, 6, 3, 2, 6, 7, 3, 6, 7, 2, 3, 7, 6, 2, 3, 2, 7, 6, 2, 2, 2,
       2, 7, 1, 3, 7, 7, 6, 6, 6, 6, 3, 1, 4, 0, 1, 4, 5, 0, 2, 0, 5, 0,
       1, 4, 5, 4, 1, 0, 5, 4, 1, 0, 1, 4, 5, 0, 2, 4, 1, 0, 5, 0, 1, 4,
       1, 4, 5, 4, 5, 4, 6, 4, 5, 4, 5, 4, 5, 4, 1, 0, 5, 0, 5, 0, 1, 4,
       2, 0, 2, 0, 1, 4, 5, 4, 1, 0, 1, 0, 1, 4, 1, 4, 5, 4, 1, 4, 1, 0,
       5, 0])
```

In [66]:
```
df1 = pd.DataFrame(scaled_data, columns = df.columns)
df1.head(10)
```

Out[66]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.0	0.019231	0.000000	0.387755
1	1.0	0.057692	0.000000	0.816327
2	0.0	0.038462	0.008493	0.051020
3	0.0	0.096154	0.008493	0.775510
4	0.0	0.250000	0.016985	0.397959
5	0.0	0.076923	0.016985	0.765306
6	0.0	0.326923	0.025478	0.051020
7	0.0	0.096154	0.025478	0.948980
8	1.0	0.884615	0.033970	0.020408
9	0.0	0.230769	0.033970	0.724490

10. Add the cluster data with the primary dataset

In [67]:
```
df1['Cluster']  = pd.Series(res)
df1.head(10)
```

Out[67]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	1.0	0.019231	0.000000	0.387755	3
1	1.0	0.057692	0.000000	0.816327	3
2	0.0	0.038462	0.008493	0.051020	7
3	0.0	0.096154	0.008493	0.775510	7
4	0.0	0.250000	0.016985	0.397959	7
5	0.0	0.076923	0.016985	0.765306	7
6	0.0	0.326923	0.025478	0.051020	6
7	0.0	0.096154	0.025478	0.948980	7
8	1.0	0.884615	0.033970	0.020408	2
9	0.0	0.230769	0.033970	0.724490	7

In [68]:
```
df1['Cluster'].unique()
```

Out[68]:
```
array([3, 7, 6, 2, 5, 1, 4, 0])
```

In [69]:
```
df1['Cluster'].value_counts()
```

Out[69]:
```
6    37
7    34
2    29
3    24
4    22
1    19
0    18
5    17
Name: Cluster, dtype: int64
```

11.Split the data into dependent and independent variables

In [70]:
```
# independent variable
X = df1.iloc[:,0:4]
X.head(10)
```

Out[70]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.0	0.019231	0.000000	0.387755
1	1.0	0.057692	0.000000	0.816327
2	0.0	0.038462	0.008493	0.051020
3	0.0	0.096154	0.008493	0.775510
4	0.0	0.250000	0.016985	0.397959
5	0.0	0.076923	0.016985	0.765306
6	0.0	0.326923	0.025478	0.051020
7	0.0	0.096154	0.025478	0.948980
8	1.0	0.884615	0.033970	0.020408
9	0.0	0.230769	0.033970	0.724490

In [71]:
```
# dependent variable
y = df1.iloc[:,4:]
y.head(10)
```

Out[71]:

	Cluster
0	3
1	3
2	7
3	7
4	7
5	7
6	6
7	7
8	2
9	7

12.Split the data into training and testing

In [72]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=
X_train.head(10)
```

Out[72]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
116	0.0	0.865385	0.424628	0.428571
67	0.0	0.961538	0.280255	0.479592
78	0.0	0.096154	0.331210	0.520408
42	1.0	0.576923	0.203822	0.357143
17	1.0	0.038462	0.050955	0.663265
5	0.0	0.076923	0.016985	0.765306
127	1.0	0.423077	0.475584	0.959184
105	0.0	0.057692	0.399151	0.418367
48	0.0	0.211538	0.212314	0.418367
66	0.0	0.480769	0.280255	0.500000

In [73]:
```
X_test.head(10)
```

Out[73]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
58	0.0	0.173077	0.263270	0.510204
40	0.0	0.903846	0.195329	0.346939
34	0.0	0.596154	0.152866	0.132653
102	1.0	0.942308	0.399151	0.591837
184	0.0	0.442308	0.713376	0.387755
198	1.0	0.269231	1.000000	0.173469
95	1.0	0.115385	0.382166	0.520408
4	0.0	0.250000	0.016985	0.397959
29	0.0	0.096154	0.118896	0.877551
168	0.0	0.346154	0.611465	0.265306

In [74]:
```
y_train.head(10)
```

Out[74]:

	Cluster
116	6
67	6
78	7
42	2
17	3
5	7
127	0
105	7
48	7
66	6

In [75]:
```
y_test.head(10)
```

Out[75]:

	Cluster
58	7
40	6
34	6
102	2
184	1
198	5
95	3
4	7
29	7
168	1

13.Build the Model

In [76]:
```
# classification algorithm
classifier_model = SVC(decision_function_shape='ovo')
```

14.Train the Model

In [77]:
```
classifier_model.fit(X_train,y_train.values.flatten())
```

Out[77]:
```
SVC(decision_function_shape='ovo')
```

15.Test the Model

In [78]:
```
pred_y = classifier_model.predict(X_test)
pred_y[0:5]
```

Out[78]:
```
array([7, 6, 6, 2, 1])
```

16. Measure the performance using Evaluation Metrics

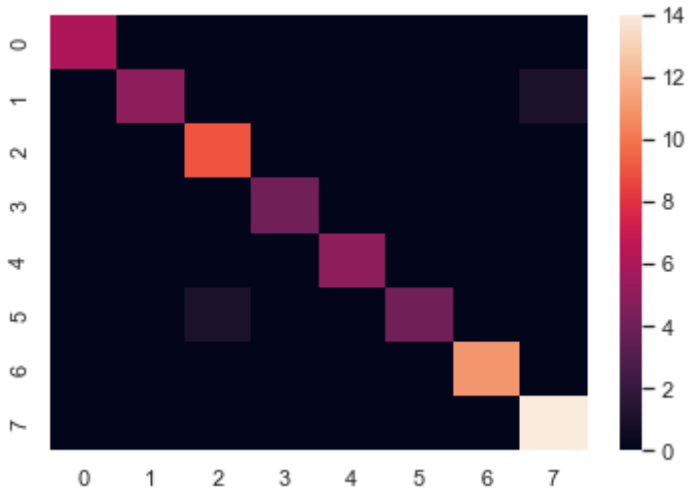
In [79]:
```
print('Classification Report: ')
print(classification_report(y_test, pred_y))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      0.83      0.91         6
           2       0.90      1.00      0.95         9
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         5
           5       1.00      0.80      0.89         5
           6       1.00      1.00      1.00        11
           7       0.93      1.00      0.97        14

    accuracy                           0.97        60
   macro avg       0.98      0.95      0.96        60
weighted avg       0.97      0.97      0.97        60
```

In [80]:
```
print('Confusion Matrix: ')
sns.heatmap(confusion_matrix(y_test,pred_y))
```

```
Confusion Matrix:
```

Out[80]:
```
<AxesSubplot:>
```



In [81]:
```
print('F1 Score: ',f1_score(y_test,pred_y, average='weighted'))
```

```
F1 Score:  0.9657091177962321
```

In [82]:
```
# Hamming loss gives the fraction of labels that are incorrectly predicted
print('Hamming Loss: ',hamming_loss(y_test,pred_y))
```

```
Hamming Loss:  0.03333333333333333
```

In [83]:
```
print('Accuracy: ',accuracy_score(y_test,pred_y))
```

```
Accuracy:  0.9666666666666667
```