Problem statement

You own the mall and want to understand the customers who can quickly converge [Target Customers] so that the insight can be given to the marketing team and plan the strategy accordingly.

In [61]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [3]:

```
data=pd.read_csv('Mall_Customers.csv')
```

In [4]:

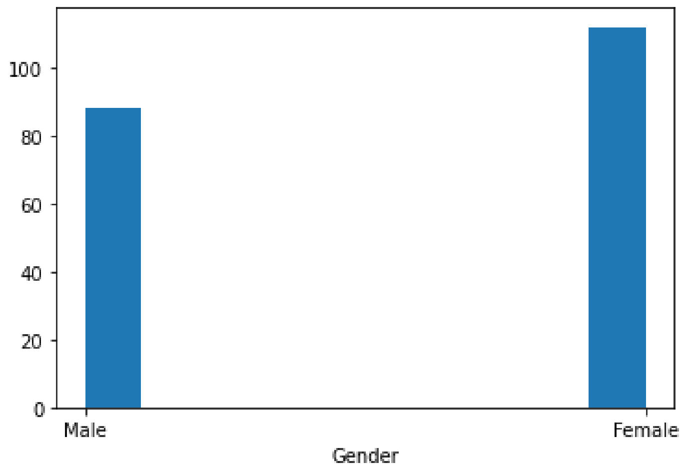
```
data.info()
```

```
<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
```

Univariate Analysis

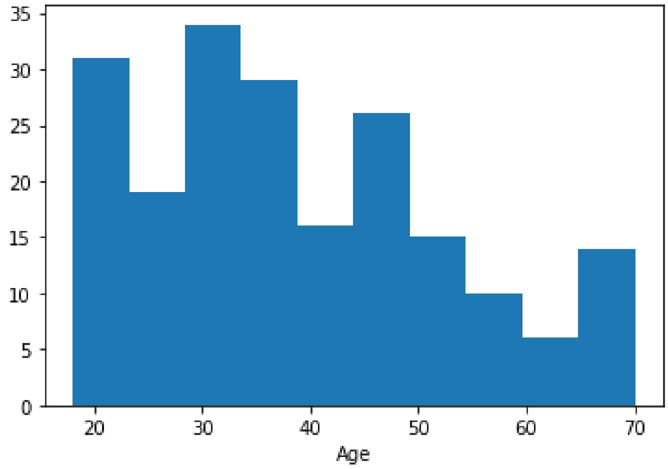
In [6]:

```
plt.hist(data['Gender']);
plt.xlabel('Gender');
```



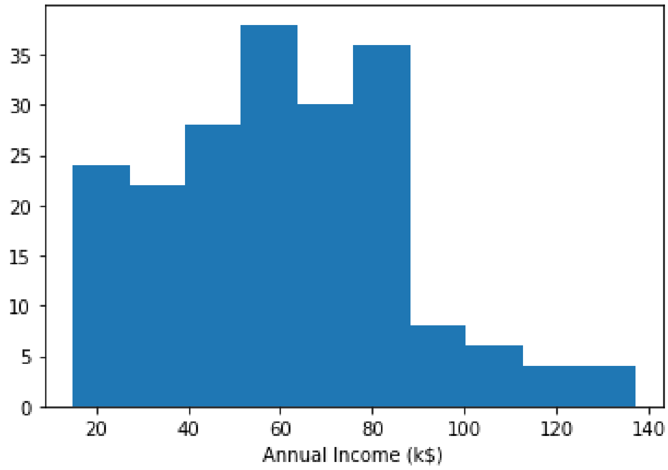
In [7]:

```
plt.hist(data['Age']);
plt.xlabel('Age');
```



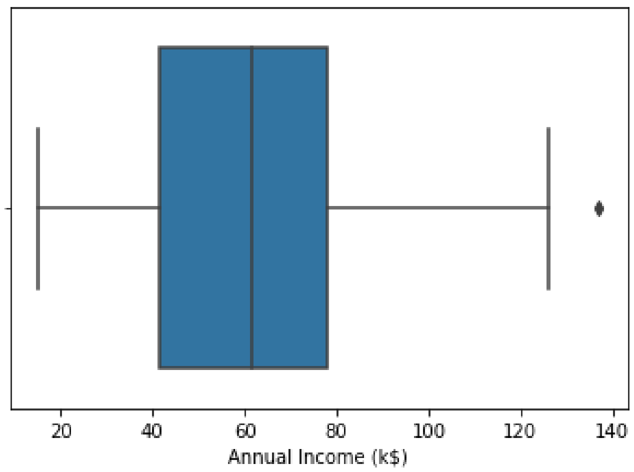
In [8]:

```
plt.hist(data['Annual Income (k$)']);
plt.xlabel('Annual Income (k$)');
```



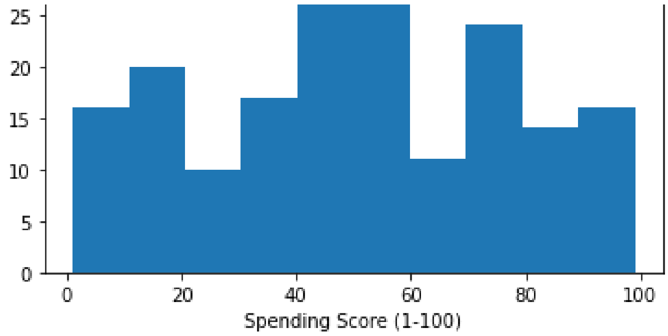
In [9]:

```
sns.boxplot(x=data['Annual Income (k$)'])
plt.xlabel('Annual Income (k$)');
```



In [10]:

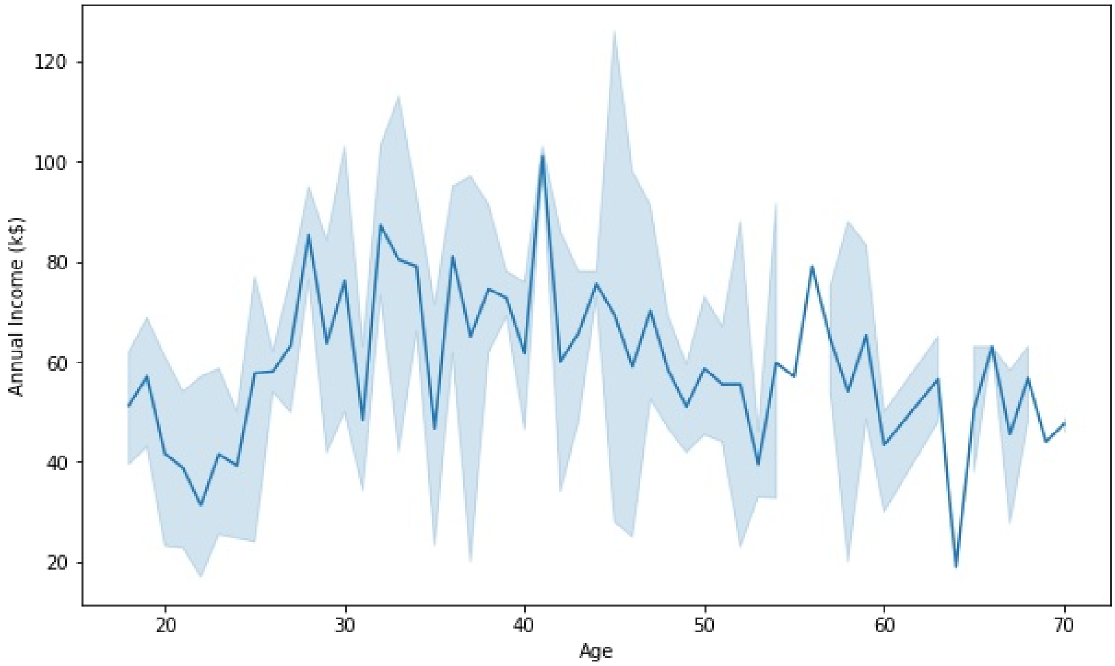
```
plt.hist(data['Spending Score (1-100)']);
plt.xlabel('Spending Score (1-100)');
```



Bivariate Analysis

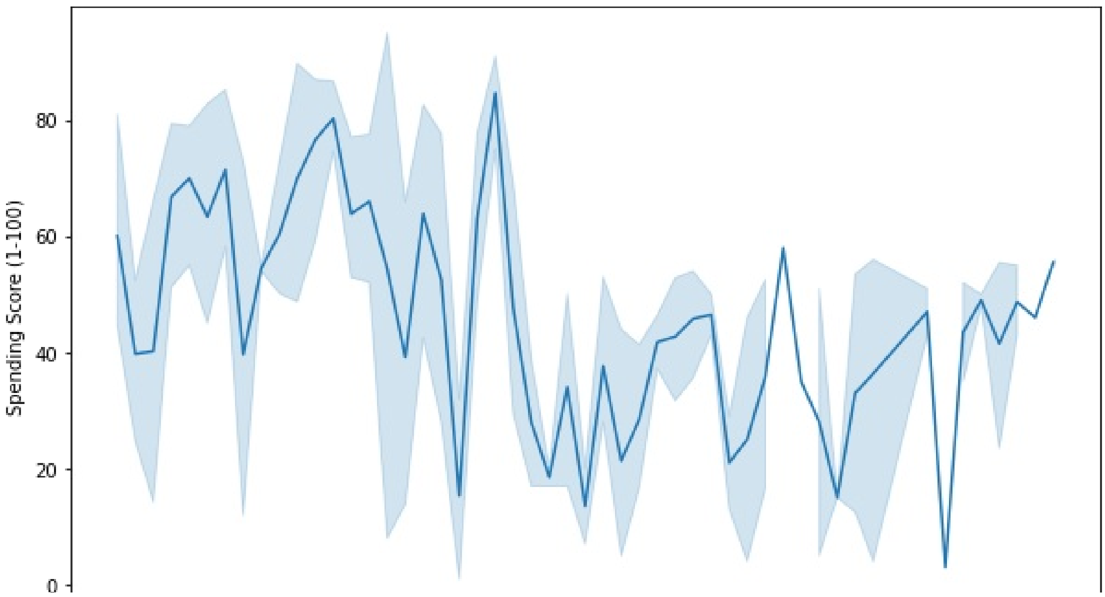
In [12]:

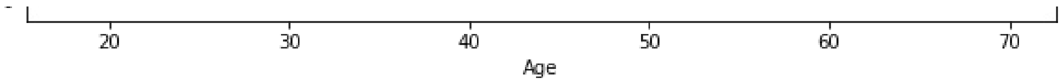
```
plt.figure(figsize=(10, 6))
sns.lineplot(x=data["Age"], y=data["Annual Income (k$)"]);
plt.xlabel('Age');
plt.ylabel('Annual Income (k$)');
```



In [13]:

```
plt.figure(figsize=(10, 6))
sns.lineplot(x=data["Age"], y=data["Spending Score (1-100)"]);
plt.xlabel('Age');
plt.ylabel('Spending Score (1-100)');
```

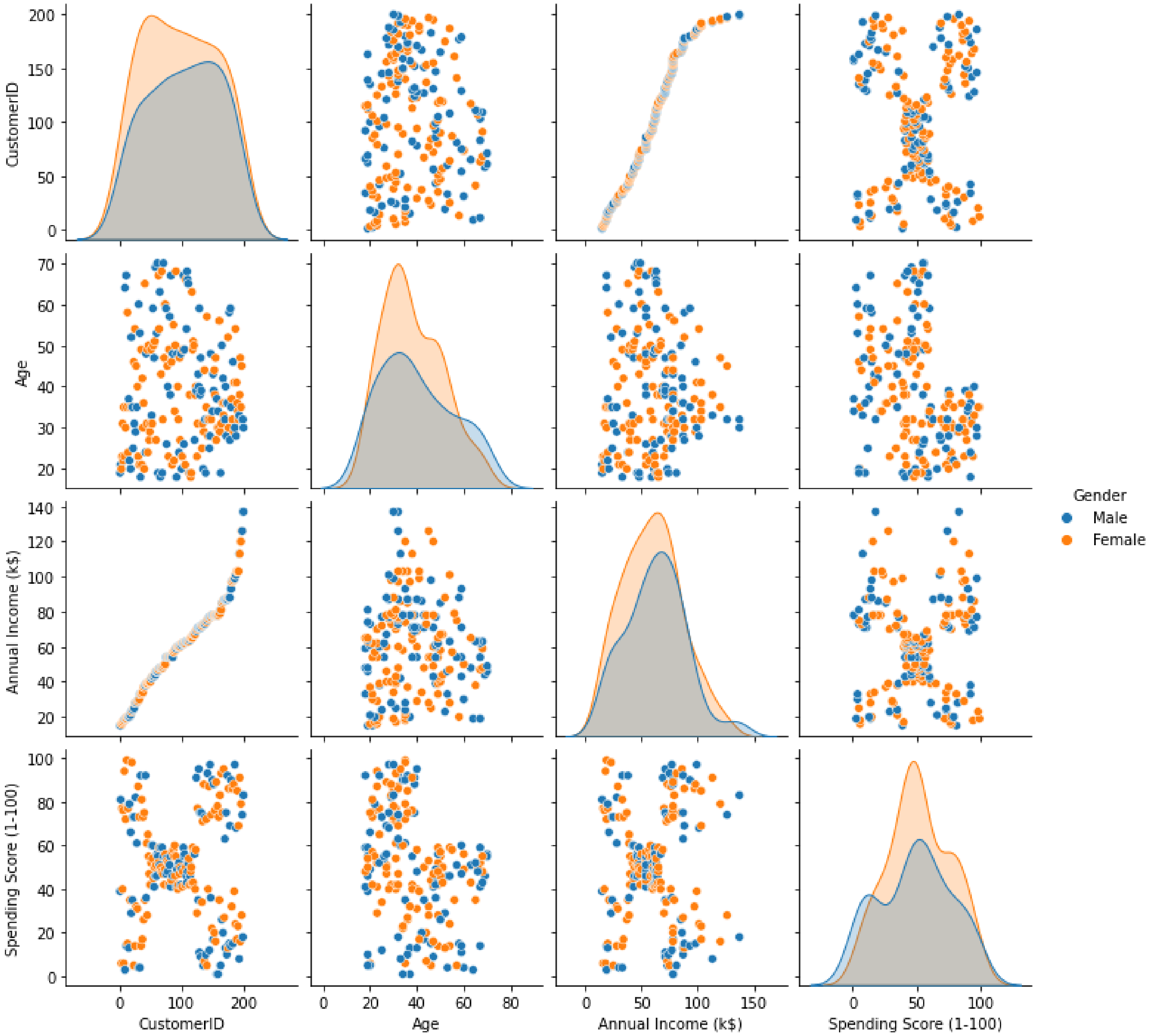




Multi-variate Analysis

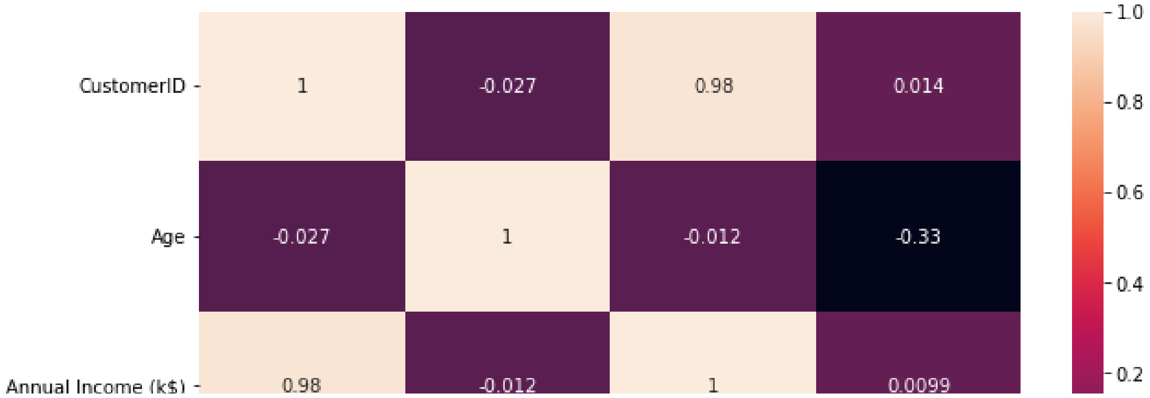
In [15]:

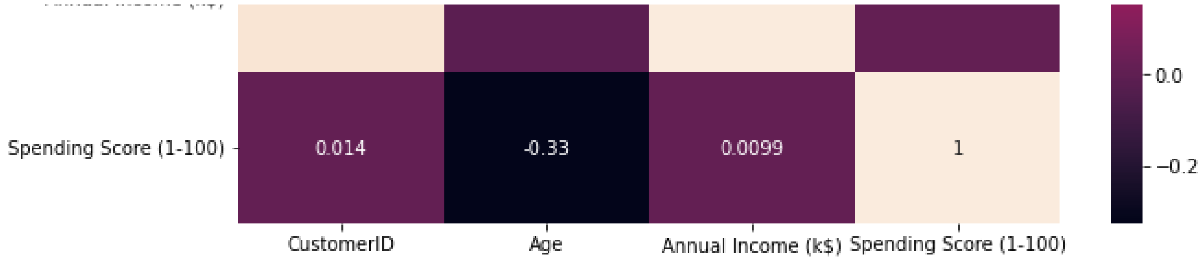
```
sns.pairplot(data, hue='Gender');
```



In [16]:

```
plt.figure(figsize=(10, 6));
sns.heatmap(data.corr(), annot=True);
```





Descriptive Statistics

In [19]:

```
data.describe()
```

Out[19]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

In [21]:

```
data.skew()
```

```
C:\Users\vkedu\AppData\Local\Temp\ipykernel_8692\1188251951.py:1: FutureWarning: Dropping
of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns before calling the
reduction.
  data.skew()
```

Out[21]:

```
CustomerID                0.000000
Age                       0.485569
Annual Income (k$)        0.321843
Spending Score (1-100)   -0.047220
dtype: float64
```

In [22]:

```
data.kurt()
```

```
C:\Users\vkedu\AppData\Local\Temp\ipykernel_8692\2907027414.py:1: FutureWarning: Dropping
of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns before calling the
reduction.
  data.kurt()
```

Out[22]:

```
CustomerID               -1.200000
Age                      -0.671573
Annual Income (k$)       -0.098487
Spending Score (1-100)   -0.826629
dtype: float64
```

In [23]:

```
data.var()
```

```
C:\Users\vkedu\AppData\Local\Temp\ipykernel_8692\445316826.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  data.var()
```

Out[23]:

```
CustomerID                3350.000000
Age                        195.133166
Annual Income (k$)         689.835578
Spending Score (1-100)     666.854271
dtype: float64
```

Handling Missing Values

In [25]:

```
data.isna().sum()
```

Out[25]:

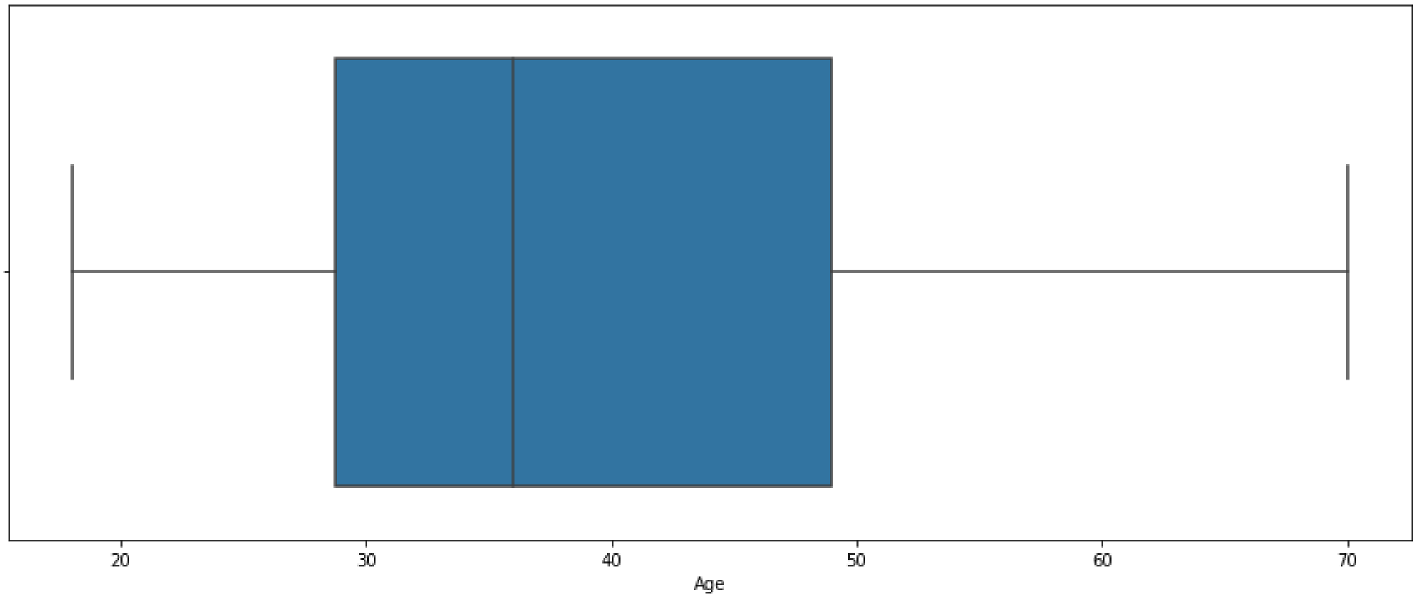
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

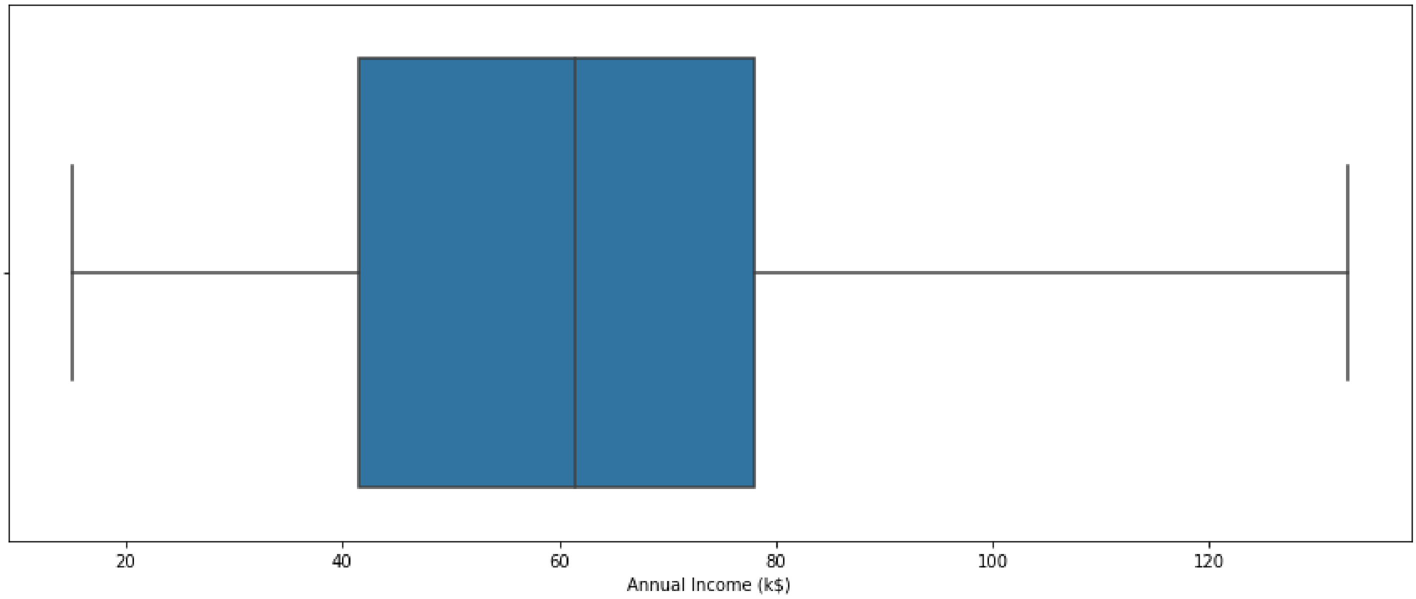
Outlier Handling

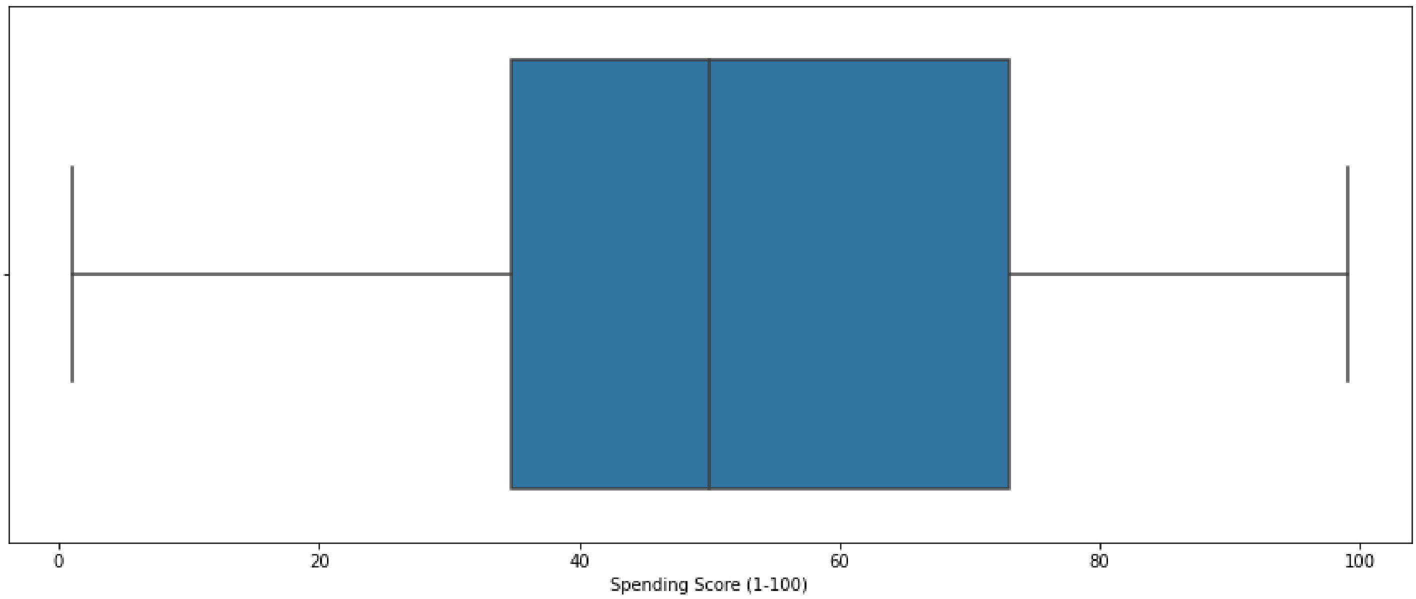
In [30]:

```
numeric_cols = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']

def boxplots(cols):
    fig, axes = plt.subplots(3, 1, figsize=(15, 20))

    t=0
    for i in range(3):
        sns.boxplot(ax=axes[i], data=data, x=cols[t])
        t+=1

    plt.show()

def Flooring_outlier(col):
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    whisker_width = 1.5
    lower_whisker = Q1 -(whisker_width*IQR)
    upper_whisker = Q3 + (whisker_width*IQR)
    data[col]=np.where(data[col]>upper_whisker,upper_whisker,np.where(data[col]<lower_whisker,lower_whisker,data[col]))

print('Before Outliers Handling')
print('='*100)
boxplots(numeric_cols)
for col in numeric_cols:
    Flooring_outlier(col)
print('\n\n\nAfter Outliers Handling')
print('='*100)
boxplots(numeric_cols)
```

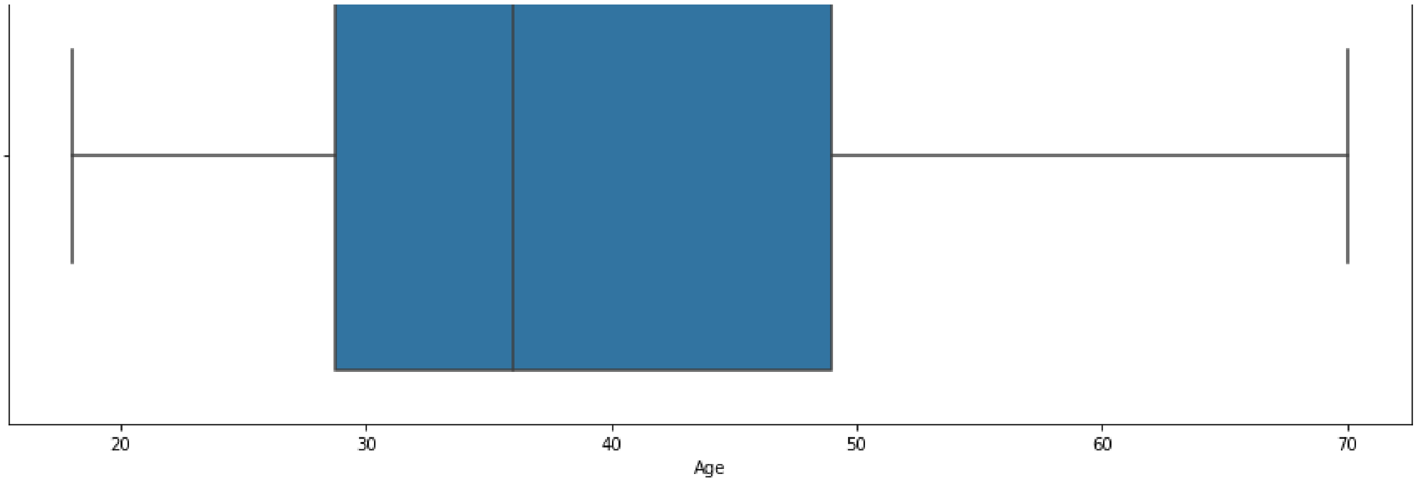
```
Before Outliers Handling
====================================================================================================
```

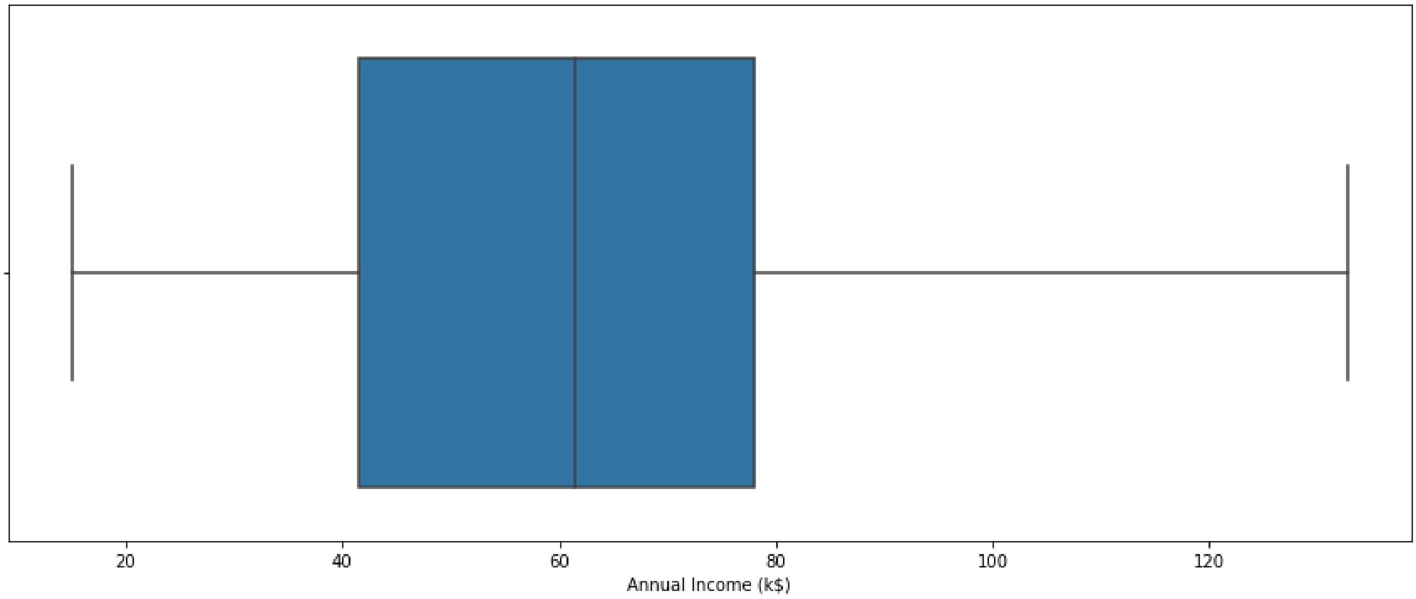


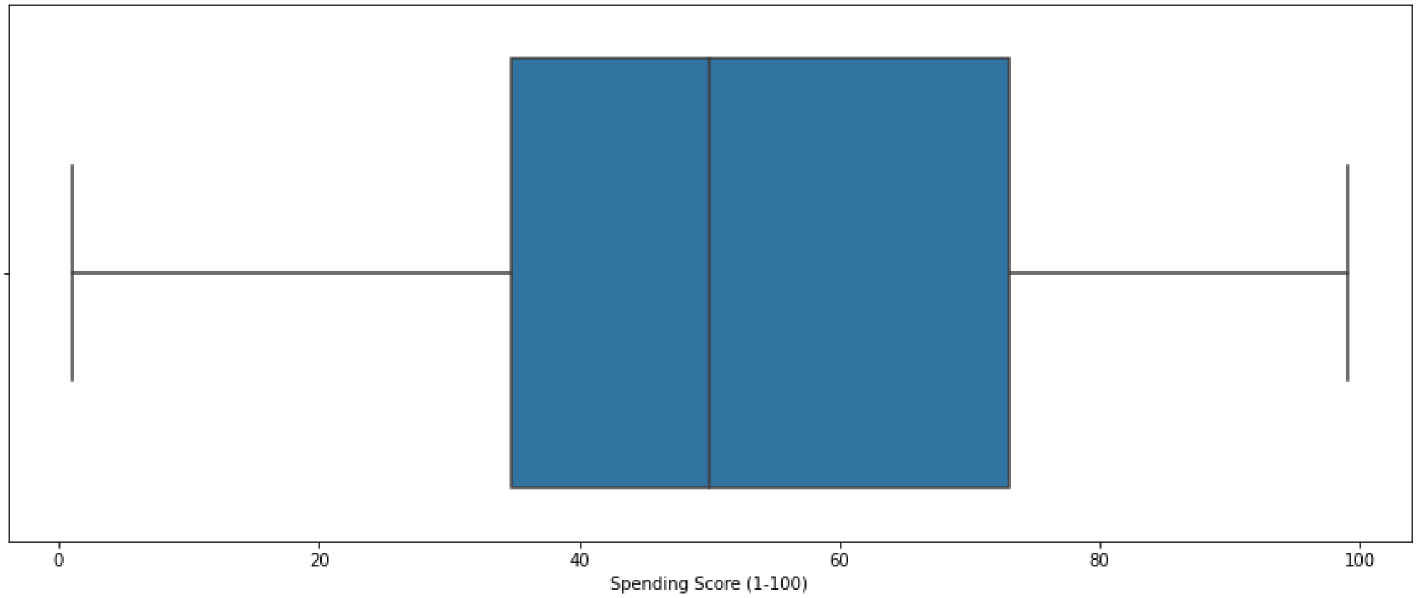





```
After Outliers Handling
==========================================================================================
```







Encode Categorical Columns

In [32]:

```
data = pd.get_dummies(data, columns = ['Gender'])
data
```

Out[32]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
0	1	19.0	15.00	39.0	0	1

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
1	2	21.0	15.00	81.0	0	1
2	3	20.0	16.00	6.0	1	0
3	4	23.0	16.00	77.0	1	0
4	5	31.0	17.00	40.0	1	0
...	...	...	...	...	...	...
195	196	35.0	120.00	79.0	1	0
196	197	45.0	126.00	28.0	1	0
197	198	32.0	126.00	74.0	0	1
198	199	32.0	132.75	18.0	0	1
199	200	30.0	132.75	83.0	0	1

200 rows × 6 columns

Standard Scaling

In [34]:

```
data = data.drop(['CustomerID'], axis=1)
data
```

Out[34]:

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
0	19.0	15.00	39.0	0	1
1	21.0	15.00	81.0	0	1
2	20.0	16.00	6.0	1	0
3	23.0	16.00	77.0	1	0
4	31.0	17.00	40.0	1	0
...	...	...	...	...	...
195	35.0	120.00	79.0	1	0
196	45.0	126.00	28.0	1	0
197	32.0	126.00	74.0	0	1
198	32.0	132.75	18.0	0	1
199	30.0	132.75	83.0	0	1

200 rows × 5 columns

In [35]:

```
cols = data.columns
cols
```

Out[35]:

```
Index(['Age', 'Annual Income (k$)', 'Spending Score (1-100)', 'Gender_Female',
       'Gender_Male'],
      dtype='object')
```

In [36]:

```
scaler = StandardScaler()
sc = scaler.fit_transform(data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']])
sc
```

Out[36]:

```
array([[-1.42456879, -1.74542941, -0.43480148],
```

```
       [-1.28103541, -1.74542941,  1.19570407],
       [-1.3528021 , -1.70708307, -1.71591298],
       [-1.13750203, -1.70708307,  1.04041783],
       [-0.56336851, -1.66873673, -0.39597992],
       [-1.20926872, -1.66873673,  1.00159627],
       [-0.27630176, -1.6303904 , -1.71591298],
       [-1.13750203, -1.6303904 ,  1.70038436],
       [ 1.80493225, -1.59204406, -1.83237767],
       [-0.6351352 , -1.59204406,  0.84631002],
       [ 2.02023231, -1.59204406, -1.4053405 ],
       [-0.27630176, -1.59204406,  1.89449216],
       [ 1.37433211, -1.55369772, -1.36651894],
       [-1.06573534, -1.55369772,  1.04041783],
       [-0.13276838, -1.55369772, -1.44416206],
       [-1.20926872, -1.55369772,  1.11806095],
       [-0.27630176, -1.51535138, -0.59008772],
       [-1.3528021 , -1.51535138,  0.61338066],
       [ 0.94373197, -1.43865871, -0.82301709],
       [-0.27630176, -1.43865871,  1.8556706 ],
       [-0.27630176, -1.40031237, -0.59008772],
       [-0.99396865, -1.40031237,  0.88513158],
       [ 0.51313183, -1.36196603, -1.75473454],
       [-0.56336851, -1.36196603,  0.88513158],
       [ 1.08726535, -1.24692702, -1.4053405 ],
       [-0.70690189, -1.24692702,  1.23452563],
       [ 0.44136514, -1.24692702, -0.7065524 ],
       [-0.27630176, -1.24692702,  0.41927286],
       [ 0.08253169, -1.20858069, -0.74537397],
       [-1.13750203, -1.20858069,  1.42863343],
       [ 1.51786549, -1.17023435, -1.7935561 ],
       [-1.28103541, -1.17023435,  0.88513158],
       [ 1.01549866, -1.05519534, -1.7935561 ],
       [-1.49633548, -1.05519534,  1.62274124],
       [ 0.7284319 , -1.05519534, -1.4053405 ],
       [-1.28103541, -1.05519534,  1.19570407],
       [ 0.22606507, -1.016849  , -1.28887582],
       [-0.6351352 , -1.016849  ,  0.88513158],
       [-0.20453507, -0.90180999, -0.93948177],
       [-1.3528021 , -0.90180999,  0.96277471],
       [ 1.87669894, -0.86346365, -0.59008772],
       [-1.06573534, -0.86346365,  1.62274124],
       [ 0.65666521, -0.82511731, -0.55126616],
       [-0.56336851, -0.82511731,  0.41927286],
       [ 0.7284319 , -0.82511731, -0.86183865],
       [-1.06573534, -0.82511731,  0.5745591 ],
       [ 0.80019859, -0.78677098,  0.18634349],
       [-0.85043527, -0.78677098, -0.12422899],
       [-0.70690189, -0.78677098, -0.3183368 ],
       [-0.56336851, -0.78677098, -0.3183368 ],
       [ 0.7284319 , -0.7100783 ,  0.06987881],
       [-0.41983513, -0.7100783 ,  0.38045129],
       [-0.56336851, -0.67173196,  0.14752193],
       [ 1.4460988 , -0.67173196,  0.38045129],
       [ 0.80019859, -0.67173196, -0.20187212],
       [ 0.58489852, -0.67173196, -0.35715836],
       [ 0.87196528, -0.63338563, -0.00776431],
       [ 2.16376569, -0.63338563, -0.16305055],
       [-0.85043527, -0.55669295,  0.03105725],
       [ 1.01549866, -0.55669295, -0.16305055],
       [ 2.23553238, -0.55669295,  0.22516505],
       [-1.42456879, -0.55669295,  0.18634349],
       [ 2.02023231, -0.51834661,  0.06987881],
       [ 1.08726535, -0.51834661,  0.34162973],
       [ 1.73316556, -0.48000028,  0.03105725],
       [-1.49633548, -0.48000028,  0.34162973],
       [ 0.29783176, -0.48000028, -0.00776431],
       [ 2.091999  , -0.48000028, -0.08540743],
       [-1.42456879, -0.48000028,  0.34162973],
       [-0.49160182, -0.48000028, -0.12422899],
       [ 2.23553238, -0.44165394,  0.18634349],
       [ 0.58489852, -0.44165394, -0.3183368 ],
       [ 1.51786549, -0.4033076 , -0.04658587],
```

```
       [ 1.51786549, -0.4033076 ,  0.22516505],
       [ 1.4460988 , -0.24992225, -0.12422899],
       [-0.92220196, -0.24992225,  0.14752193],
       [ 0.44136514, -0.24992225,  0.10870037],
       [ 0.08253169, -0.24992225, -0.08540743],
       [-1.13750203, -0.24992225,  0.06987881],
       [ 0.7284319 , -0.24992225, -0.3183368 ],
       [ 1.30256542, -0.24992225,  0.03105725],
       [-0.06100169, -0.24992225,  0.18634349],
       [ 2.02023231, -0.24992225, -0.35715836],
       [ 0.51313183, -0.24992225, -0.24069368],
       [-1.28103541, -0.24992225,  0.26398661],
       [ 0.65666521, -0.24992225, -0.16305055],
       [ 1.15903204, -0.13488324,  0.30280817],
       [-1.20926872, -0.13488324,  0.18634349],
       [-0.34806844, -0.0965369 ,  0.38045129],
       [ 0.80019859, -0.0965369 , -0.16305055],
       [ 2.091999  , -0.05819057,  0.18634349],
       [-1.49633548, -0.05819057, -0.35715836],
       [ 0.65666521, -0.01984423, -0.04658587],
       [ 0.08253169, -0.01984423, -0.39597992],
       [-0.49160182, -0.01984423, -0.3183368 ],
       [-1.06573534, -0.01984423,  0.06987881],
       [ 0.58489852, -0.01984423, -0.12422899],
       [-0.85043527, -0.01984423, -0.00776431],
       [ 0.65666521,  0.01850211, -0.3183368 ],
       [-1.3528021 ,  0.01850211, -0.04658587],
       [-1.13750203,  0.05684845, -0.35715836],
       [ 0.7284319 ,  0.05684845, -0.08540743],
       [ 2.02023231,  0.05684845,  0.34162973],
       [-0.92220196,  0.05684845,  0.18634349],
       [ 0.7284319 ,  0.05684845,  0.22516505],
       [-1.28103541,  0.05684845, -0.3183368 ],
       [ 1.94846562,  0.09519478, -0.00776431],
       [ 1.08726535,  0.09519478, -0.16305055],
       [ 2.091999  ,  0.09519478, -0.27951524],
       [ 1.94846562,  0.09519478, -0.08540743],
       [ 1.87669894,  0.09519478,  0.06987881],
       [-1.42456879,  0.09519478,  0.14752193],
       [-0.06100169,  0.13354112, -0.3183368 ],
       [-1.42456879,  0.13354112, -0.16305055],
       [-1.49633548,  0.17188746, -0.08540743],
       [-1.42456879,  0.17188746, -0.00776431],
       [ 1.73316556,  0.17188746, -0.27951524],
       [ 0.7284319 ,  0.17188746,  0.34162973],
       [ 0.87196528,  0.24858013, -0.27951524],
       [ 0.80019859,  0.24858013,  0.26398661],
       [-0.85043527,  0.24858013,  0.22516505],
       [-0.06100169,  0.24858013, -0.39597992],
       [ 0.08253169,  0.32527281,  0.30280817],
       [ 0.010765  ,  0.32527281,  1.58391968],
       [-1.13750203,  0.36361914, -0.82301709],
       [-0.56336851,  0.36361914,  1.04041783],
       [ 0.29783176,  0.40196548, -0.59008772],
       [ 0.08253169,  0.40196548,  1.73920592],
       [ 1.4460988 ,  0.40196548, -1.52180518],
       [-0.06100169,  0.40196548,  0.96277471],
       [ 0.58489852,  0.40196548, -1.5994483 ],
       [ 0.010765  ,  0.40196548,  0.96277471],
       [-0.99396865,  0.44031182, -0.62890928],
       [-0.56336851,  0.44031182,  0.80748846],
       [-1.3528021 ,  0.47865816, -1.75473454],
       [-0.70690189,  0.47865816,  1.46745499],
       [ 0.36959845,  0.47865816, -1.67709142],
       [-0.49160182,  0.47865816,  0.88513158],
       [-1.42456879,  0.51700449, -1.56062674],
       [-0.27630176,  0.51700449,  0.84631002],
       [ 1.30256542,  0.55535083, -1.75473454],
       [-0.49160182,  0.55535083,  1.6615628 ],
       [-0.77866858,  0.59369717, -0.39597992],
       [-0.49160182,  0.59369717,  1.42863343],
       [-0.99396865,  0.6320435 , -1.48298362],
```

```
       [-0.77866858,  0.6320435 ,  1.81684904],
       [ 0.65666521,  0.6320435 , -0.55126616],
       [-0.49160182,  0.6320435 ,  0.92395314],
       [-0.34806844,  0.67038984, -1.09476801],
       [-0.34806844,  0.67038984,  1.54509812],
       [ 0.29783176,  0.67038984, -1.28887582],
       [ 0.010765  ,  0.67038984,  1.46745499],
       [ 0.36959845,  0.67038984, -1.17241113],
       [-0.06100169,  0.67038984,  1.00159627],
       [ 0.58489852,  0.67038984, -1.32769738],
       [-0.85043527,  0.67038984,  1.50627656],
       [-0.13276838,  0.67038984, -1.91002079],
       [-0.6351352 ,  0.67038984,  1.07923939],
       [-0.34806844,  0.67038984, -1.91002079],
       [-0.6351352 ,  0.67038984,  0.88513158],
       [ 1.23079873,  0.70873618, -0.59008772],
       [-0.70690189,  0.70873618,  1.27334719],
       [-1.42456879,  0.78542885, -1.75473454],
       [-0.56336851,  0.78542885,  1.6615628 ],
       [ 0.80019859,  0.9388142 , -0.93948177],
       [-0.20453507,  0.9388142 ,  0.96277471],
       [ 0.22606507,  0.97716054, -1.17241113],
       [-0.41983513,  0.97716054,  1.73920592],
       [-0.20453507,  1.01550688, -0.90066021],
       [-0.49160182,  1.01550688,  0.49691598],
       [ 0.08253169,  1.01550688, -1.44416206],
       [-0.77866858,  1.01550688,  0.96277471],
       [-0.20453507,  1.01550688, -1.56062674],
       [-0.20453507,  1.01550688,  1.62274124],
       [ 0.94373197,  1.05385321, -1.44416206],
       [-0.6351352 ,  1.05385321,  1.38981187],
       [ 1.37433211,  1.05385321, -1.36651894],
       [-0.85043527,  1.05385321,  0.72984534],
       [ 1.4460988 ,  1.2455849 , -1.4053405 ],
       [-0.27630176,  1.2455849 ,  1.54509812],
       [-0.13276838,  1.39897025, -0.7065524 ],
       [-0.49160182,  1.39897025,  1.38981187],
       [ 0.51313183,  1.43731659, -1.36651894],
       [-0.70690189,  1.43731659,  1.46745499],
       [ 0.15429838,  1.47566292, -0.43480148],
       [-0.6351352 ,  1.47566292,  1.81684904],
       [ 1.08726535,  1.5523556 , -1.01712489],
       [-0.77866858,  1.5523556 ,  0.69102378],
       [ 0.15429838,  1.62904827, -1.28887582],
       [-0.20453507,  1.62904827,  1.35099031],
       [-0.34806844,  1.62904827, -1.05594645],
       [-0.49160182,  1.62904827,  0.72984534],
       [-0.41983513,  2.01251165, -1.63826986],
       [-0.06100169,  2.01251165,  1.58391968],
       [ 0.58489852,  2.28093601, -1.32769738],
       [-0.27630176,  2.28093601,  1.11806095],
       [ 0.44136514,  2.51101403, -0.86183865],
       [-0.49160182,  2.51101403,  0.92395314],
       [-0.49160182,  2.76985181, -1.25005425],
       [-0.6351352 ,  2.76985181,  1.27334719]])
```

In [37]:

```
data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = sc
data
```

Out[37]:

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
0	-1.424569	-1.745429	-0.434801	0	1
1	-1.281035	-1.745429	1.195704	0	1
2	-1.352802	-1.707083	-1.715913	1	0
3	-1.137502	-1.707083	1.040418	1	0
4	-0.563369	-1.668737	0.395980	1	0

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
4	-0.563369	-1.668737	-0.395980	1	0
...	...	...	...	...	...
195	-0.276302	2.280936	1.118061	1	0
196	0.441365	2.511014	-0.861839	1	0
197	-0.491602	2.511014	0.923953	0	1
198	-0.491602	2.769852	-1.250054	0	1
199	-0.635135	2.769852	1.273347	0	1

200 rows × 5 columns

Clustering

In [39]:

```
TWSS = []
k = list(range(2,13))

for i in k:
    kmeans = KMeans(n_clusters = i , init = 'k-means++')
    kmeans.fit(data)
    TWSS.append(kmeans.inertia_)

TWSS
```

Out[39]:

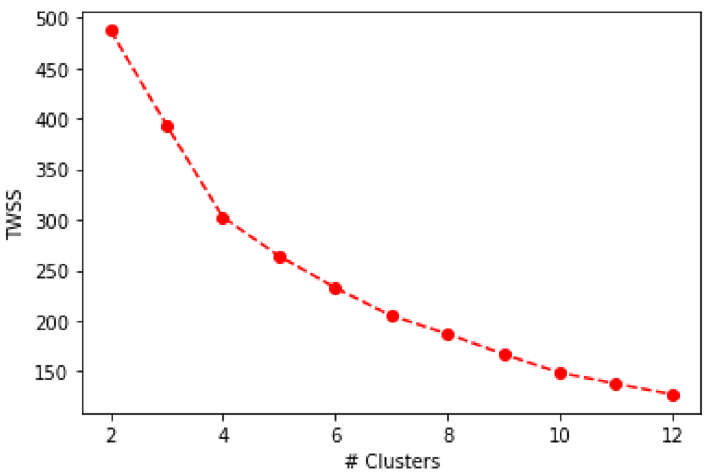
```
[487.6586717274497,
 393.64978299868295,
 302.7542334541678,
 264.4053832507914,
 233.03386823093447,
 205.10124246839075,
 187.39595996551753,
 167.0398397965737,
 148.88636365194475,
 137.930832935199,
 127.33536880175306]
```

In [40]:

```
plt.plot(k, TWSS, 'ro--')
plt.xlabel('# Clusters')
plt.ylabel('TWSS')
```

Out[40]:

```
Text(0, 0.5, 'TWSS')
```



In [41]:

```
model = KMeans(n_clusters = 5)
```

```
model.fit(data)
```

Out[41]:

```
KMeans(n_clusters=5)
```

Add the Cluster data with Primary dataset

In [44]:

```
mb = pd.Series(model.labels_)
data['Cluster'] = mb
data
```

Out[44]:

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male	Cluster
0	-1.424569	-1.745429	-0.434801	0	1	3
1	-1.281035	-1.745429	1.195704	0	1	3
2	-1.352802	-1.707083	-1.715913	1	0	0
3	-1.137502	-1.707083	1.040418	1	0	3
4	-0.563369	-1.668737	-0.395980	1	0	0
...	...	...	...	...	...	...
195	-0.276302	2.280936	1.118061	1	0	2
196	0.441365	2.511014	-0.861839	1	0	4
197	-0.491602	2.511014	0.923953	0	1	2
198	-0.491602	2.769852	-1.250054	0	1	4
199	-0.635135	2.769852	1.273347	0	1	2

200 rows × 6 columns

In [45]:

```
data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = scaler.inverse_transform
(data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']])
data
```

Out[45]:

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male	Cluster
0	19.0	15.00	39.0	0	1	3
1	21.0	15.00	81.0	0	1	3
2	20.0	16.00	6.0	1	0	0
3	23.0	16.00	77.0	1	0	3
4	31.0	17.00	40.0	1	0	0
...	...	...	...	...	...	...
195	35.0	120.00	79.0	1	0	2
196	45.0	126.00	28.0	1	0	4
197	32.0	126.00	74.0	0	1	2
198	32.0	132.75	18.0	0	1	4
199	30.0	132.75	83.0	0	1	2

200 rows × 6 columns

In [46]:

```
mb=pd.Series(model.labels_)
data
```

Out[46]:

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male	Cluster
0	19.0	15.00	39.0	0	1	3
1	21.0	15.00	81.0	0	1	3
2	20.0	16.00	6.0	1	0	0
3	23.0	16.00	77.0	1	0	3
4	31.0	17.00	40.0	1	0	0
...	...	...	...	...	...	...
195	35.0	120.00	79.0	1	0	2
196	45.0	126.00	28.0	1	0	4
197	32.0	126.00	74.0	0	1	2
198	32.0	132.75	18.0	0	1	4
199	30.0	132.75	83.0	0	1	2

200 rows × 6 columns

Split Data Into Dependent & Independent Features

In [48]:

```
X=data.drop('Cluster',axis=1)
Y=data['Cluster']
X, Y
```

Out[48]:

```
(       Age  Annual Income (k$)  Spending Score (1-100)  Gender_Female  \
 0     19.0               15.00                    39.0              0
 1     21.0               15.00                    81.0              0
 2     20.0               16.00                     6.0              1
 3     23.0               16.00                    77.0              1
 4     31.0               17.00                    40.0              1
 ..     ...                 ...                     ...            ...
 195   35.0              120.00                    79.0              1
 196   45.0              126.00                    28.0              1
 197   32.0              126.00                    74.0              0
 198   32.0              132.75                    18.0              0
 199   30.0              132.75                    83.0              0

      Gender_Male
 0              1
 1              1
 2              0
 3              0
 4              0
 ..           ...
 195            0
 196            0
 197            1
 198            1
 199            1

 [200 rows x 5 columns],
 0      3
 1      3
 2      0
 3      3
 4      0
       ..
 195    2
```

```
195    2
196    4
197    2
198    4
199    2
Name: Cluster, Length: 200, dtype: int32)
```

Split the data into Training And Testing Data

In [50]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[50]:

```
((160, 5), (40, 5), (160,), (40,))
```

Train Model & Evaluate

In [52]:

```
model=DecisionTreeClassifier()
model.fit(X_train,Y_train)
```

Out[52]:

```
DecisionTreeClassifier()
```

Evaluate

In [54]:

```
model.score(X_train, Y_train)
```

Out[54]:

```
1.0
```

In [55]:

```
model.score(X_test, Y_test)
```

Out[55]:

```
0.975
```

In [56]:

```
Y_pred = model.predict(X_test)
```

In [57]:

```
accuracy_score(Y_pred, Y_test)
```

Out[57]:

```
0.975
```

In [58]:

```
print(classification_report(Y_pred, Y_test))
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        11
           1       1.00      1.00      1.00        10
           2       1.00      1.00      1.00         5
           3       1.00      1.00      1.00         3
```

```
           4       1.00      0.91      0.95        11

    accuracy                           0.97        40
   macro avg       0.98      0.98      0.98        40
weighted avg       0.98      0.97      0.97        40
```

In [59]:

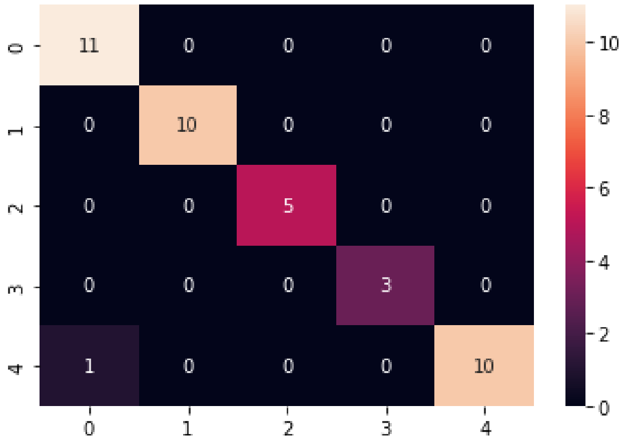
```
cm = confusion_matrix(Y_pred, Y_test)
cm
```

Out[59]:

```
array([[11,  0,  0,  0,  0],
       [ 0, 10,  0,  0,  0],
       [ 0,  0,  5,  0,  0],
       [ 0,  0,  0,  3,  0],
       [ 1,  0,  0,  0, 10]], dtype=int64)
```

In [60]:

```
sns.heatmap(cm, annot=True);
```



In [ ]: