assignment-5

October 5, 2023

1 Kaggle Connection & DataFrame setup

```
[1049]: !pip install -q kaggle
```

```
[1050]: !mkdir ~/.kaggle
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

```
[1051]: !cp kaggle.json ~/.kaggle
```

```
cp: cannot stat 'kaggle.json': No such file or directory
```

```
[1052]: ! kaggle datasets download -d vjchoudhary7/
          ↪customer-segmentation-tutorial-in-python
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix
this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
customer-segmentation-tutorial-in-python.zip: Skipping, found more recently
modified local copy (use --force to force download)
```

```
[1053]: !unzip /content/customer-segmentation-tutorial-in-python.zip
```

```
Archive:  /content/customer-segmentation-tutorial-in-python.zip
  inflating: Mall_Customers.csv
```

2 Pre-Processing

```
[1054]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
[1055]: df = pd.read_csv('./Mall_Customers.csv')
        df.head()
```

```
[1055]:    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
```

[1056]:
```
df.describe()
```

[1056]:
```
       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
```

[1057]:
```
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   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
```

[1058]:
```
df.isnull().values.any()
```

[1058]:
```
False
```

[1059]:
```
df.shape
```

[1059]:
```
(200, 5)
```

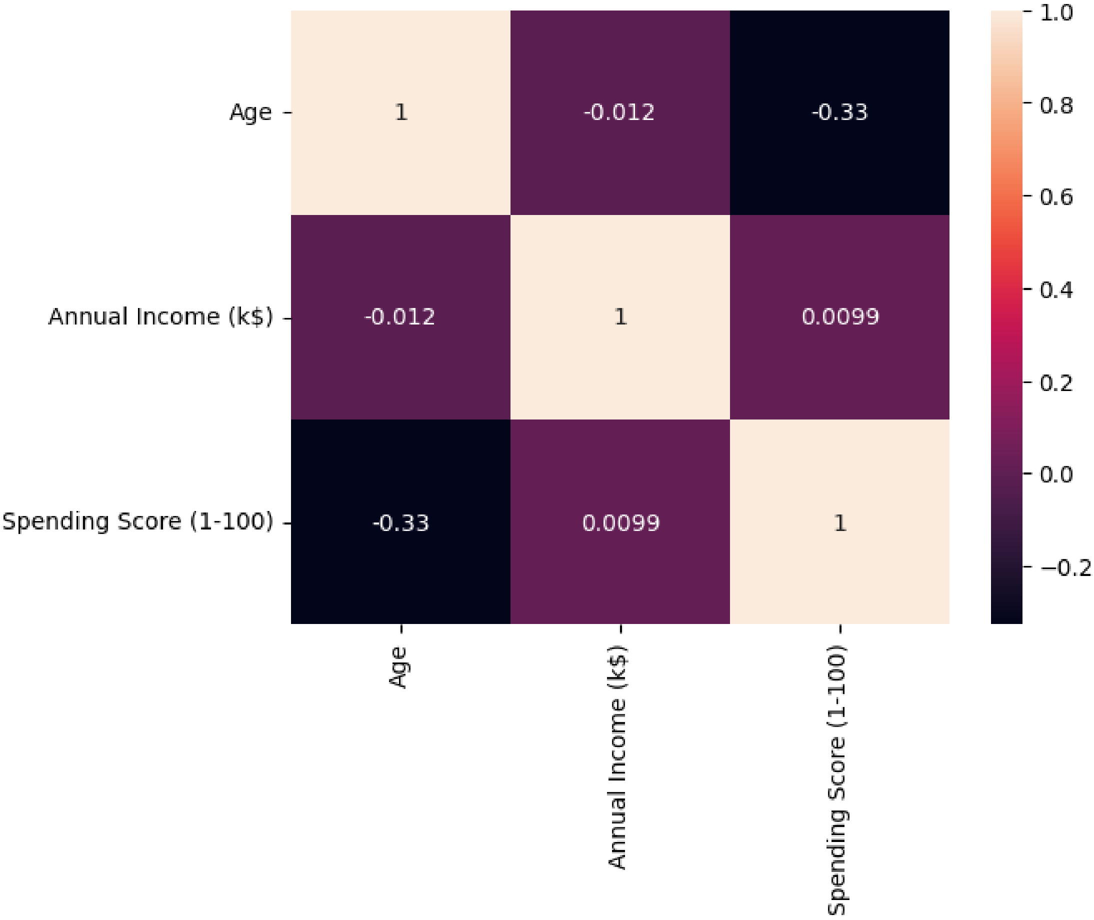
[1060]:
```
# Dropping 'CustomerID' as it has no impact or connection to dataset or data␣
 ↪values
df.drop(['CustomerID'], axis=1, inplace=True)
```

[1061]:
```
sns.heatmap(df.corr(), annot=True)
```

```
<ipython-input-1061-6dc1c4c1753e>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
```

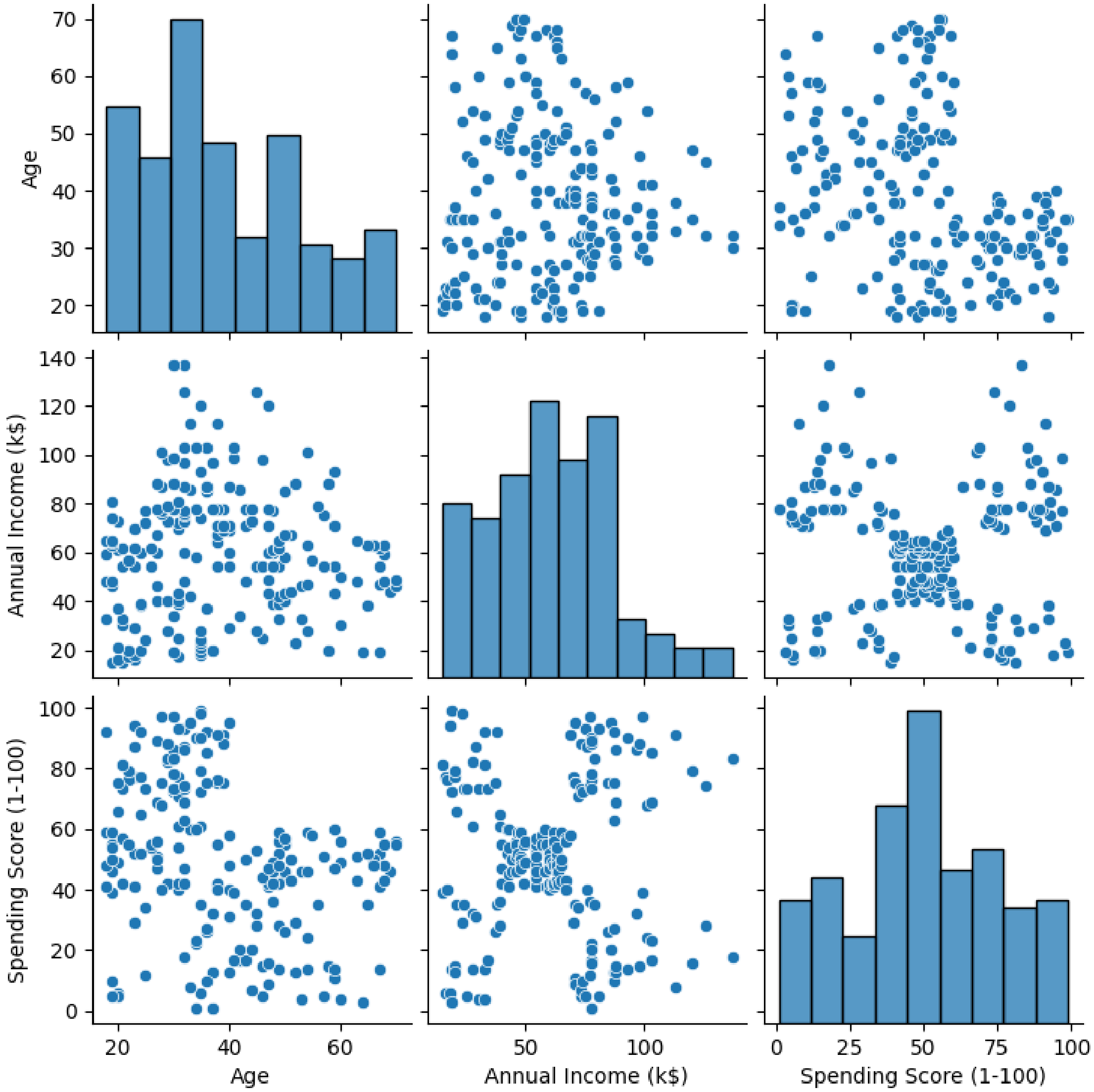
```
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr(), annot=True)
```

```
[1061]: <Axes: >
```



```
[1062]: sns.pairplot(df)
```

```
[1062]: <seaborn.axisgrid.PairGrid at 0x7dadda28db40>
```



3 Converting Categorical Data (Columns) to Numerical

[1063]:

```
df['Gender'].value_counts()
```

[1063]:

```
Female    112
Male       88
Name: Gender, dtype: int64
```

[1064]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[1065]:
```
# Label Encoding 'Gender' column
# '1' == 'Male' && 0 == 'Female'
df['Gender'] = le.fit_transform(df.Gender)
```

[1066]:
```
df.head()
```

[1066]:
```
   Gender  Age  Annual Income (k$)  Spending Score (1-100)
0       1   19                  15                      39
1       1   21                  15                      81
2       0   20                  16                       6
3       0   23                  16                      77
4       0   31                  17                      40
```

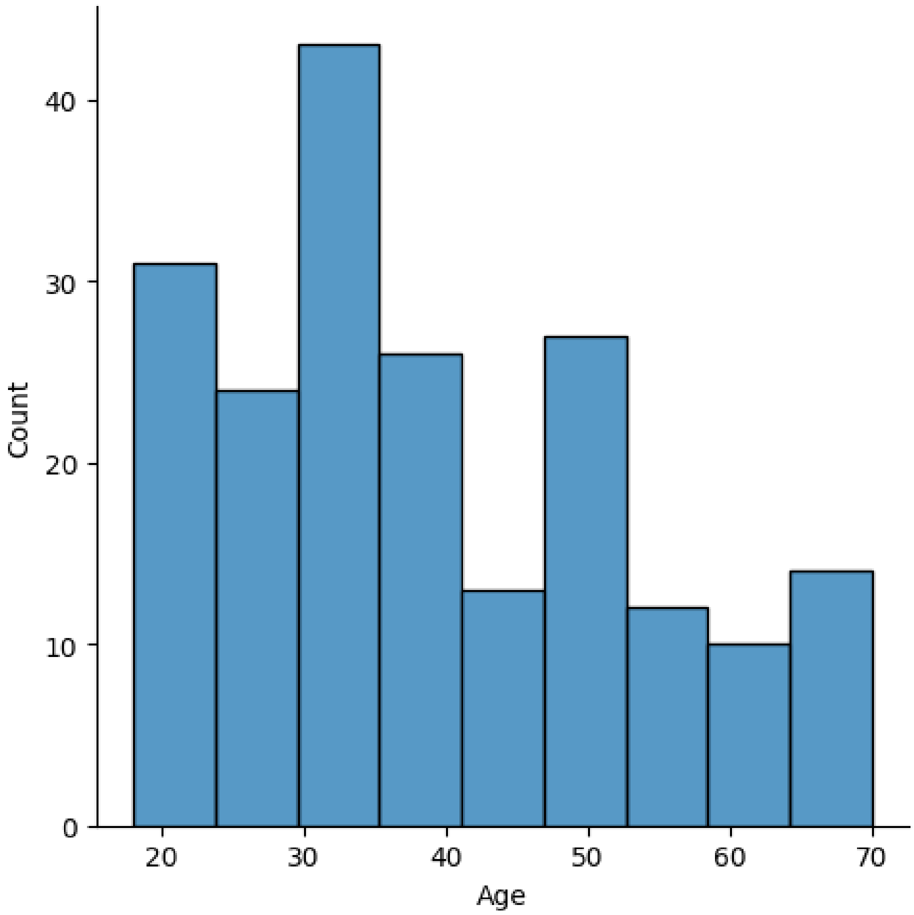
[1067]:
```
df.info()
```

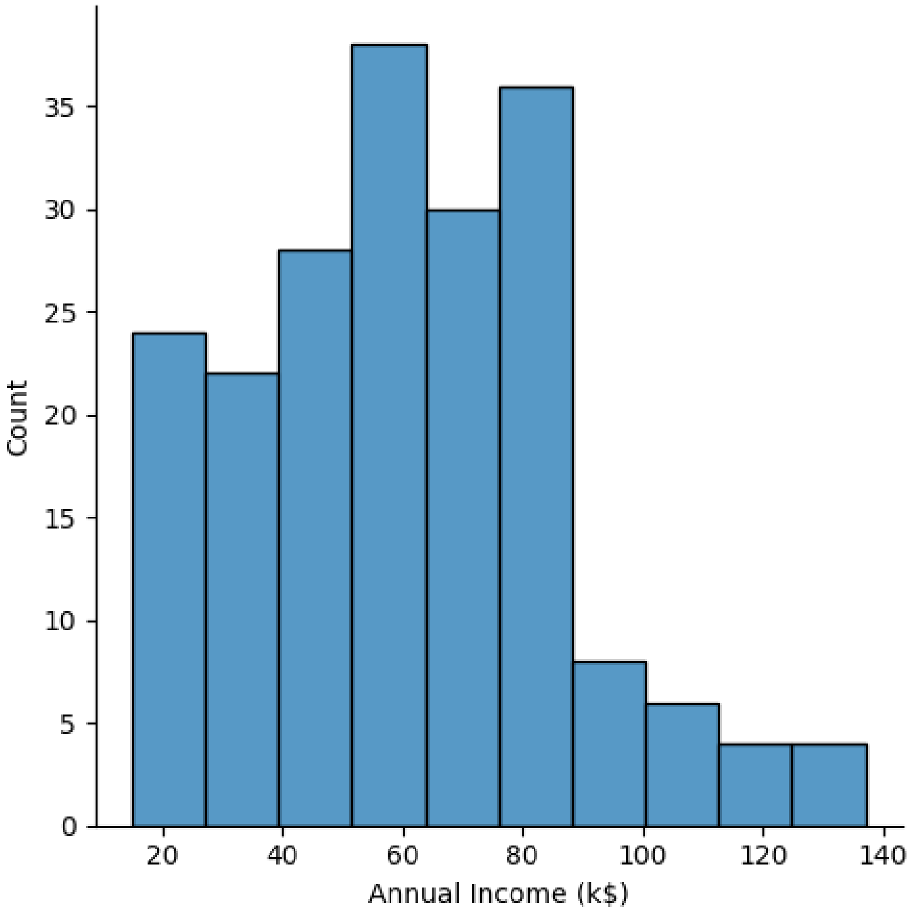
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Gender                  200 non-null    int64
 1   Age                     200 non-null    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
```

4 Data Analysis, Outlier Detection & Outlier Elimination

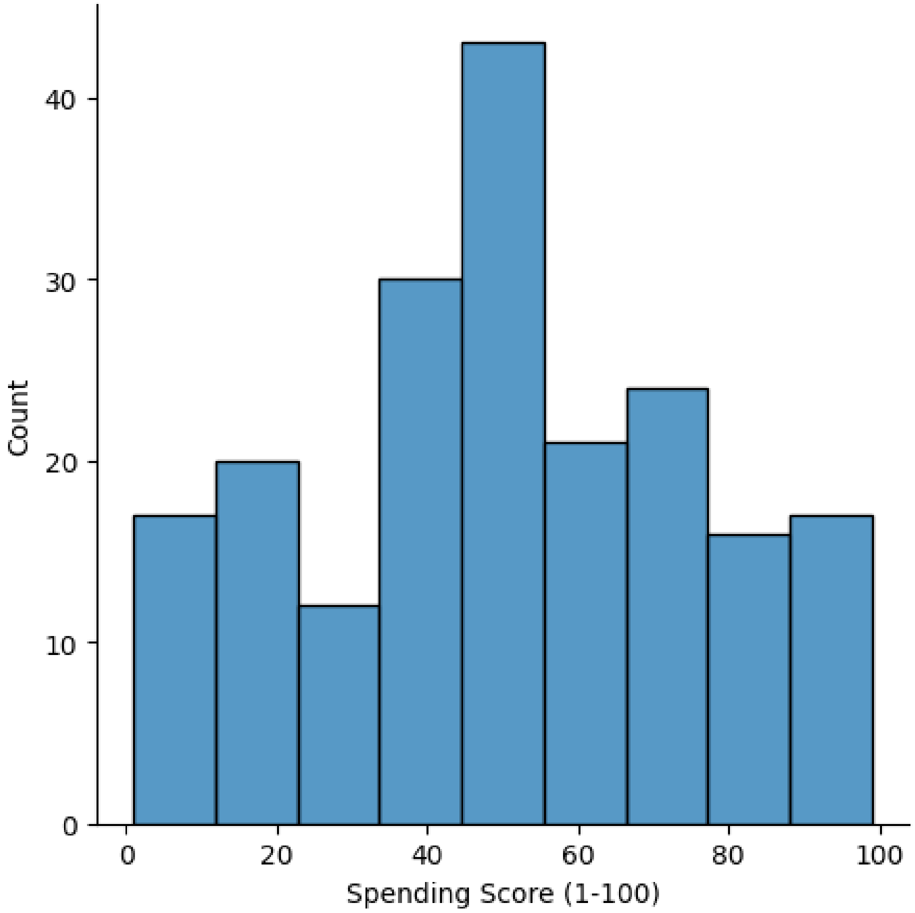
[1068]:
```
sns.displot(df['Age'])
sns.displot(df['Annual Income (k$)'])
sns.displot(df['Spending Score (1-100)'])
```

[1068]:
```
<seaborn.axisgrid.FacetGrid at 0x7dadd99c8370>
```


Count
0
10
20
30
40
20
30
40
50
60
70
Age



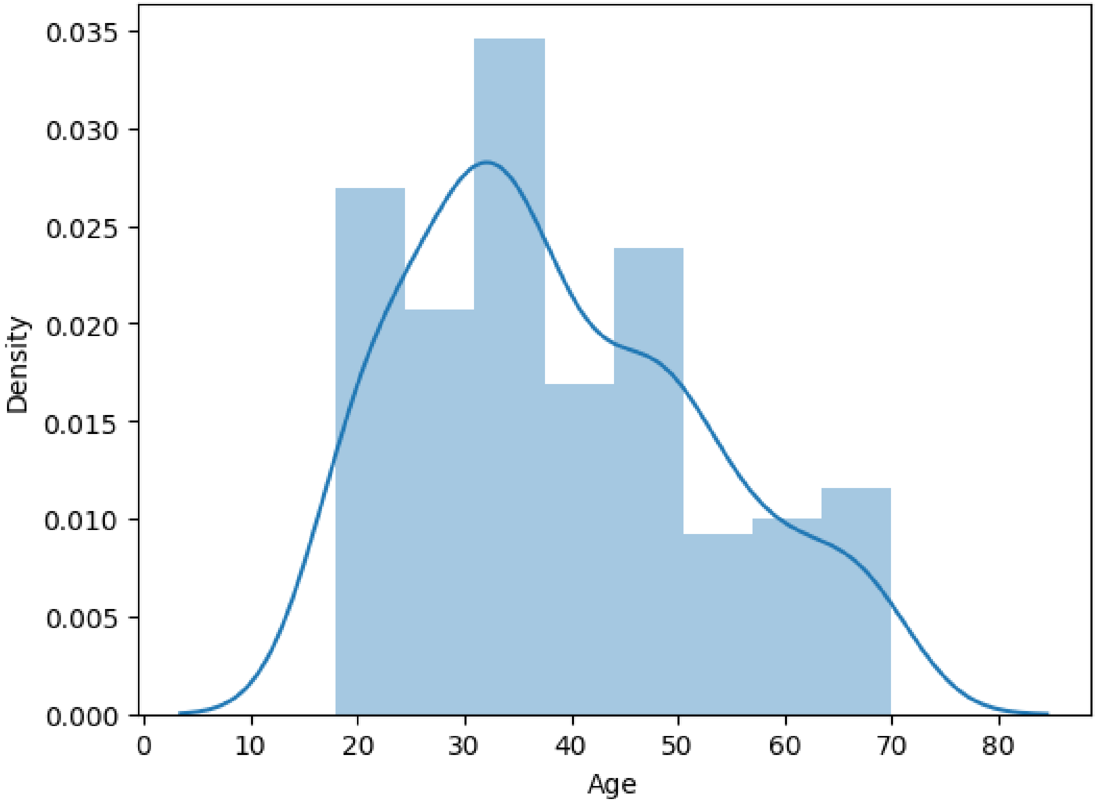
Count
0
5
10
15
20
25
30
35
20
40
60
80
100
120
140
Annual Income (k$)



```
[1069]: sns.distplot(df['Age'])
```

```
<ipython-input-1069-0fafe04ea3f6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Age'])
```

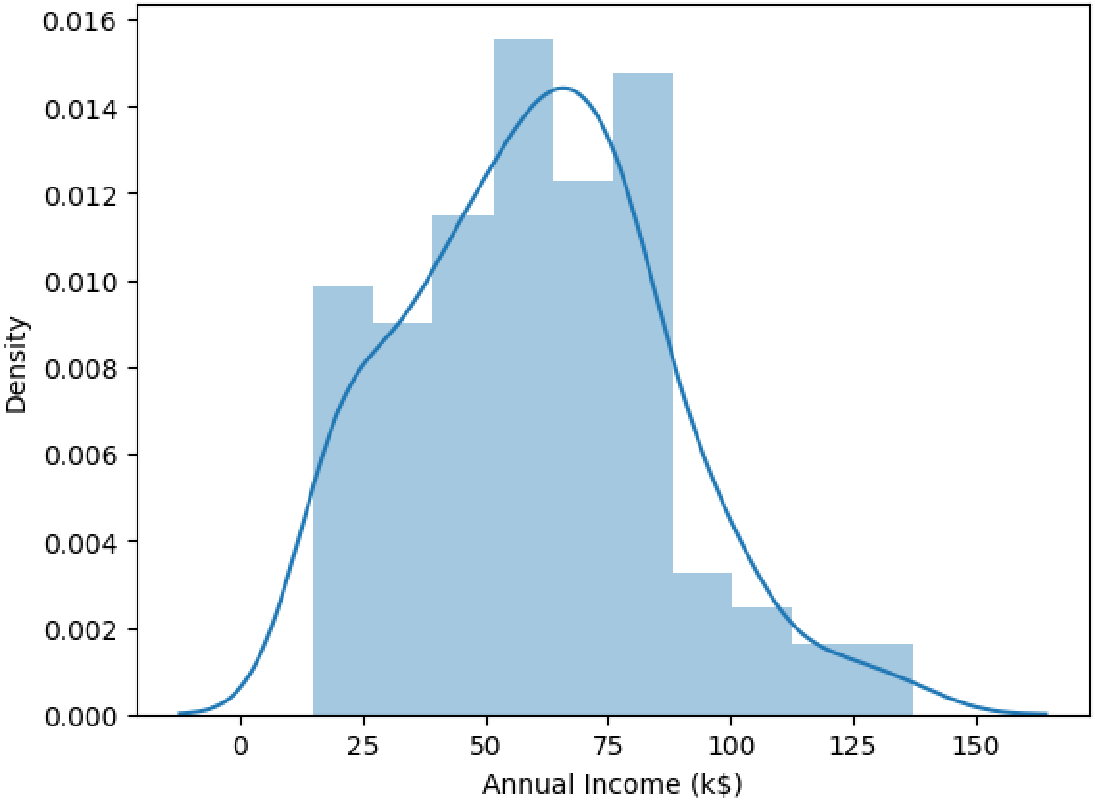
```
[1069]: <Axes: xlabel='Age', ylabel='Density'>
```



```
[1070]: sns.distplot(df['Annual Income (k$)'])
```

```
<ipython-input-1070-5c9bfeb4bab1>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Annual Income (k$)'])
```

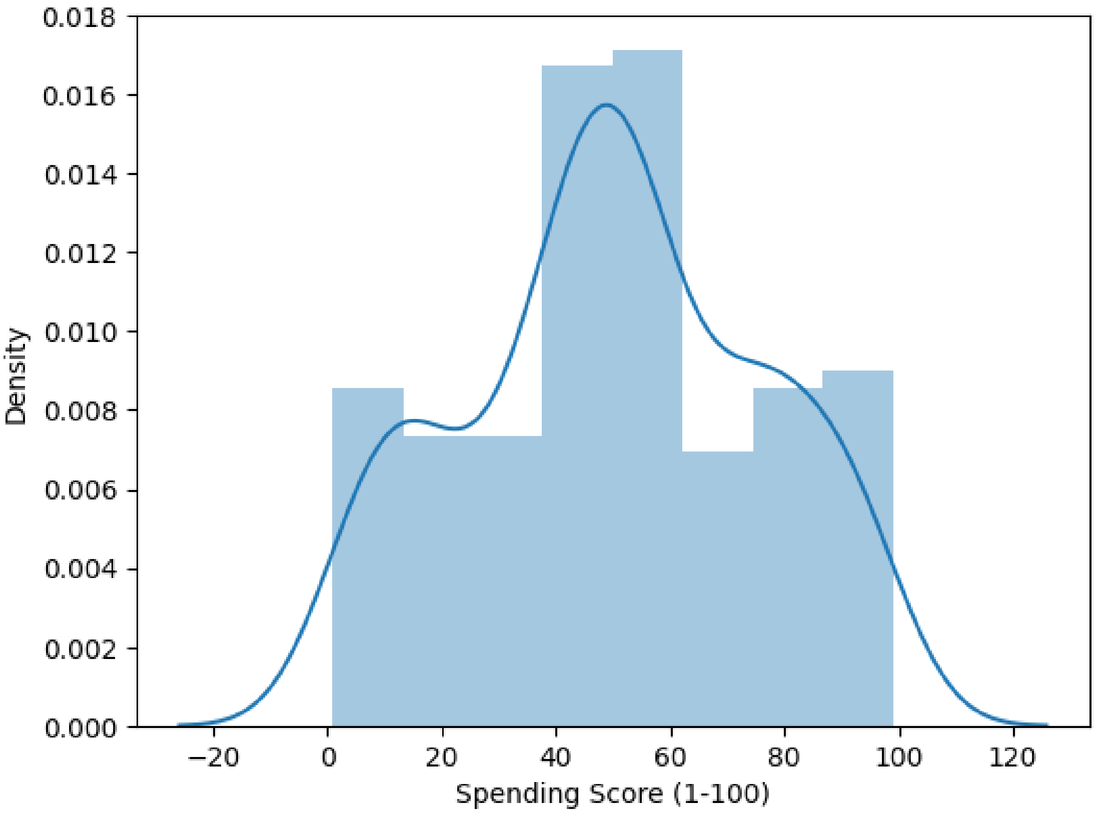
```
[1070]: <Axes: xlabel='Annual Income (k$)', ylabel='Density'>
```



[1071]:
```
sns.distplot(df['Spending Score (1-100)'])
```

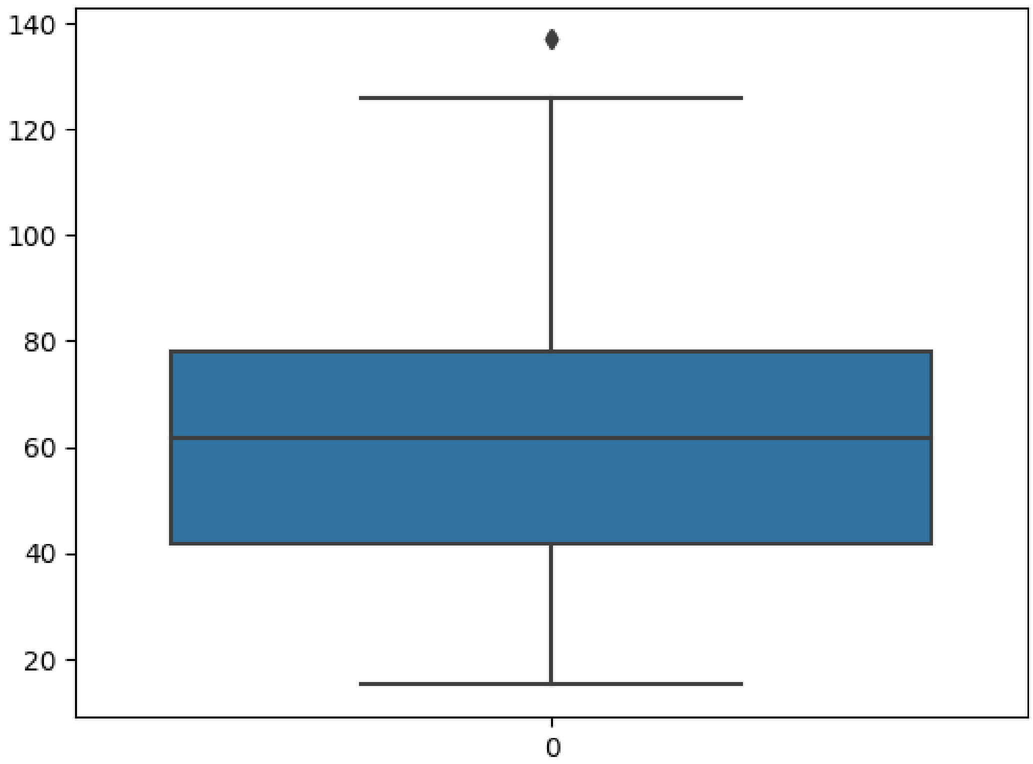
```
<ipython-input-1071-beed7b40d5ab>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Spending Score (1-100)'])
```

[1071]:
```
<Axes: xlabel='Spending Score (1-100)', ylabel='Density'>
```



```
[1072]: sns.boxplot(df['Annual Income (k$)'])
```

```
[1072]: <Axes: >
```



[1073]:
```
Q1 = df['Annual Income (k$)'].quantile(0.25)
Q3 = df['Annual Income (k$)'].quantile(0.75)
```

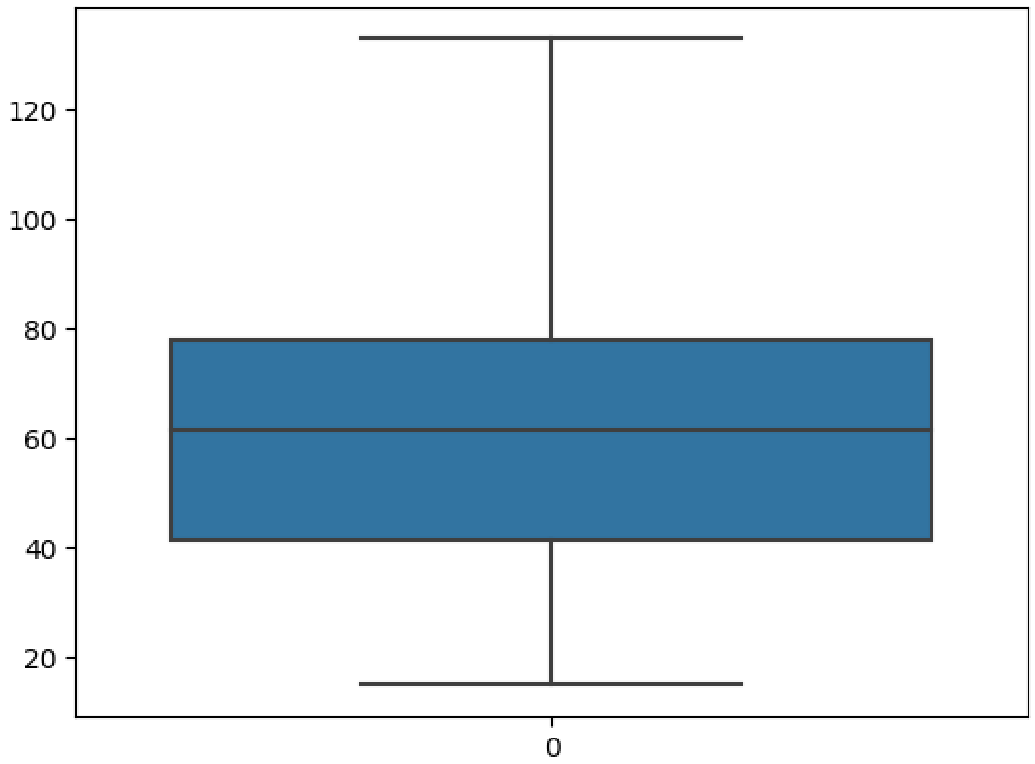
[1074]:
```
IQR = Q3 - Q1
whisker_width = 1.5
```

[1075]:
```
lower_whisker = Q1 - (whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
```

[1076]:
```
df['Annual Income (k$)'] = np.where(df['Annual Income (k$)'] > upper_whisker,␣
 ↪upper_whisker, np.where(df['Annual Income (k$)'] < lower_whisker,␣
 ↪lower_whisker, df['Annual Income (k$)']))
```

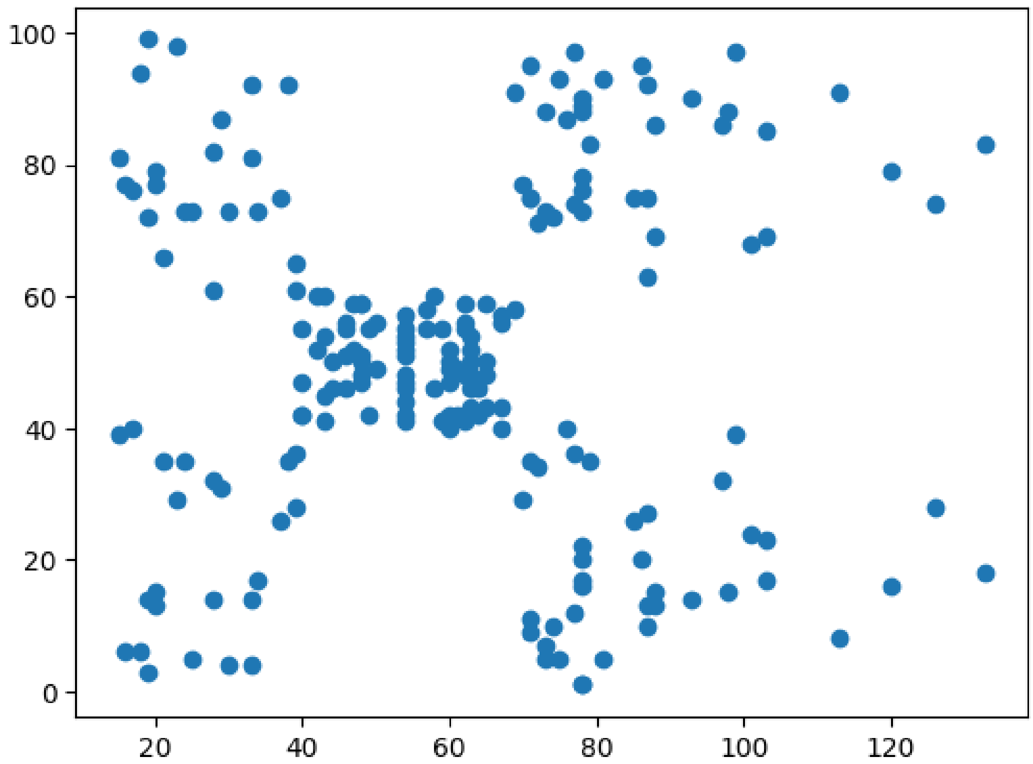
[1077]:
```
sns.boxplot(df['Annual Income (k$)'])
```

[1077]:
```
<Axes: >
```



```
[1078]: plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'])
```

```
[1078]: <matplotlib.collections.PathCollection at 0x7dadd943d780>
```



5 Finding Elbow Point (Possible 'K' value)

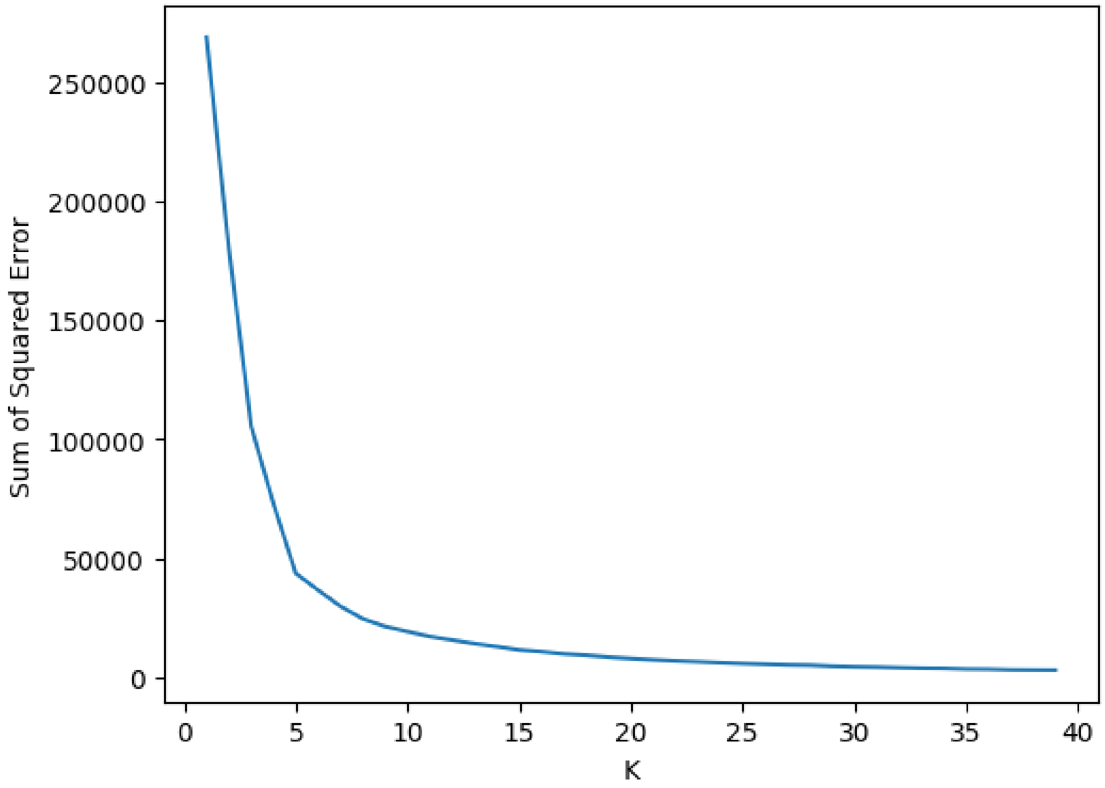
[1079]:
```
from sklearn.cluster import KMeans
```

[ ]:
```
k_rng = range(1,40)
sse = []

for k in k_rng:
  km = KMeans(n_clusters=k)
  km.fit(df[['Annual Income (k$)', 'Spending Score (1-100)']])
  sse.append(km.inertia_)
```

[ ]:
```
sse
```

[1082]:
```
plt.xlabel('K')
plt.ylabel('Sum of Squared Error')
plt.plot(k_rng, sse)
```

[1082]:
```
[<matplotlib.lines.Line2D at 0x7dadd94d1300>]
```



6 K-Means

[1083]:

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

[1083]:

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

[1084]:

```
z = kmeans.fit_predict(df[['Annual Income (k$)','Spending Score (1-100)']])
z
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[1084]:

```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 2,
       4, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 1, 2, 1, 0, 1, 0, 1,
       2, 1, 0, 1, 0, 1, 0, 1, 0, 1, 2, 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], dtype=int32)
```

[1085]:

```
df_2 = df
```

[1086]:

```
df_2['Cluster'] = z
df_2.head()
```

[1086]:

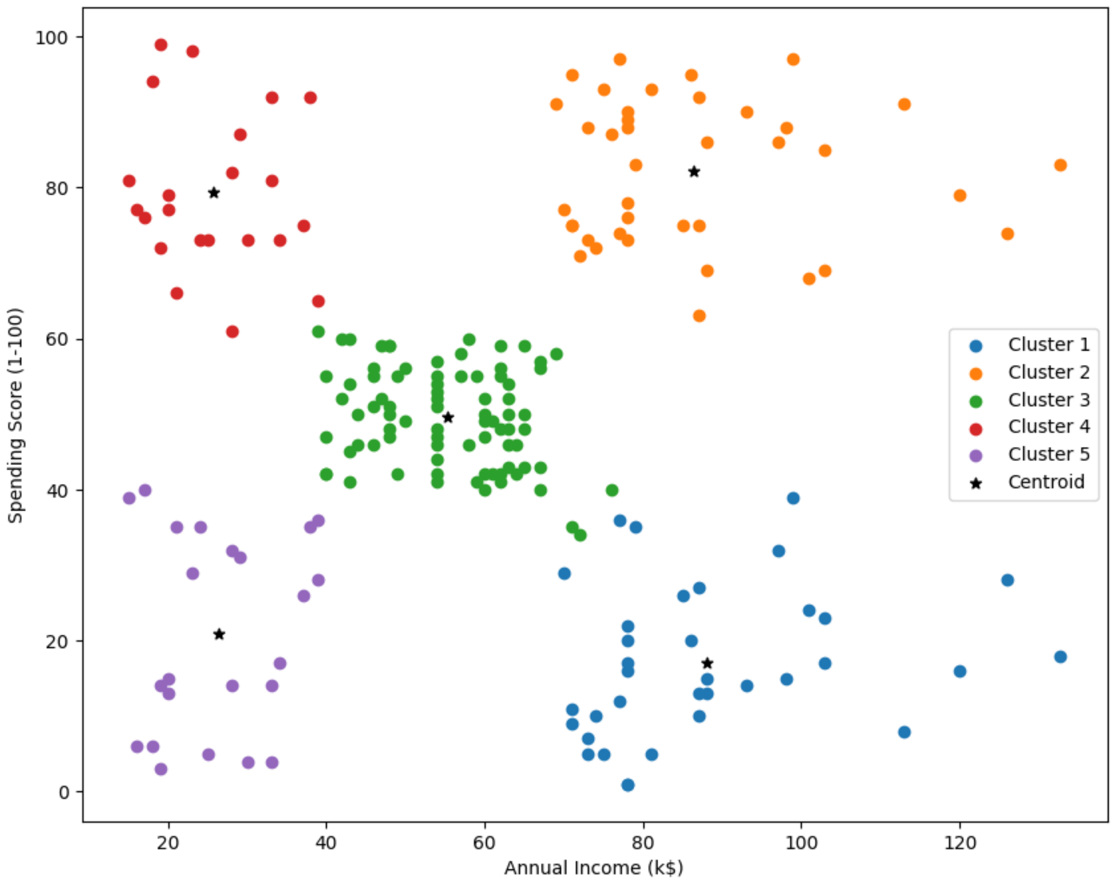
```
   Gender  Age  Annual Income (k$)  Spending Score (1-100)  Cluster
0       1   19                15.0                      39        4
1       1   21                15.0                      81        3
2       0   20                16.0                       6        4
3       0   23                16.0                      77        3
4       0   31                17.0                      40        4
```

[1087]:

```
plt.figure(figsize=(10,8))

df1 = df_2[df_2.Cluster==0]
df2 = df_2[df_2.Cluster==1]
df3 = df_2[df_2.Cluster==2]
df4 = df_2[df_2.Cluster==3]
df5 = df_2[df_2.Cluster==4]


plt.scatter(df1['Annual Income (k$)'], df1['Spending Score (1-100)'],␣
 ↪label='Cluster 1')
plt.scatter(df2['Annual Income (k$)'], df2['Spending Score (1-100)'],␣
 ↪label='Cluster 2')
plt.scatter(df3['Annual Income (k$)'], df3['Spending Score (1-100)'],␣
 ↪label='Cluster 3')
plt.scatter(df4['Annual Income (k$)'], df4['Spending Score (1-100)'],␣
 ↪label='Cluster 4')
plt.scatter(df5['Annual Income (k$)'], df5['Spending Score (1-100)'],␣
 ↪label='Cluster 5')

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],␣
 ↪color='black', marker='*', label='Centroid')

plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
```

[1087]:

```
<matplotlib.legend.Legend at 0x7dadd9366620>
```



7 Scaling & Train - Test Split

```
[1088]: from sklearn.cluster import KMeans
        from sklearn.preprocessing import MinMaxScaler
        from sklearn.model_selection import train_test_split
```

```
[1089]: df.drop(['Cluster'], axis=1, inplace=True)
```

```
[1090]: df = pd.DataFrame(MinMaxScaler().fit_transform(df), columns=df.columns)
```

```
[1091]: # X = df.drop('Spending Score (1-100)', axis=1)
         # y = df['Spending Score (1-100)']
```

```
[1092]: # xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3)
```

```
[1098]: kmeans = KMeans(n_clusters=5, init = 'k-means++',random_state=0)
```

```
[1099]: kmeans.fit(df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

```
[1099]: KMeans(n_clusters=5, random_state=0)
```

```
[1100]: X.head()
```

```
[1100]:    Gender       Age  Annual Income (k$)
        0     1.0  0.019231            0.000000
        1     1.0  0.057692            0.000000
        2     0.0  0.038462            0.008493
        3     0.0  0.096154            0.008493
        4     0.0  0.250000            0.016985
```

```
[1101]: kmeans.predict(df)
```

```
[1101]: array([3, 3, 2, 1, 1, 1, 2, 1, 4, 1, 4, 1, 2, 1, 4, 3, 2, 3, 4, 1, 4, 3,
               2, 3, 2, 3, 2, 3, 2, 1, 4, 1, 4, 3, 2, 1, 2, 1, 2, 1, 2, 3, 4, 1,
               2, 1, 2, 1, 1, 1, 2, 3, 1, 4, 2, 4, 2, 4, 1, 4, 4, 3, 2, 2, 4, 3,
               2, 2, 3, 1, 4, 2, 2, 2, 4, 3, 2, 4, 1, 2, 4, 3, 4, 2, 1, 4, 2, 1,
               1, 2, 2, 3, 4, 2, 1, 3, 2, 1, 4, 3, 1, 2, 4, 3, 4, 1, 2, 4, 4, 4,
               4, 1, 2, 3, 1, 1, 2, 2, 2, 2, 3, 2, 1, 3, 1, 1, 0, 3, 4, 3, 0, 3,
               1, 1, 0, 1, 2, 3, 0, 1, 2, 3, 1, 1, 0, 3, 4, 1, 2, 3, 0, 3, 2, 1,
               2, 1, 0, 1, 0, 1, 2, 1, 0, 1, 0, 1, 0, 1, 2, 3, 0, 3, 0, 3, 2, 1,
               0, 3, 0, 3, 2, 1, 0, 1, 2, 3, 2, 3, 2, 1, 2, 1, 0, 1, 2, 1, 2, 3,
               0, 3], dtype=int32)
```

8 Prediction

```
[1102]: kmeans.predict([[1, 19, 15, 81]])[0]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KMeans was fitted with feature names
  warnings.warn(
```

```
[1102]: 3
```