Assignment -4

Python Programming

Assignment Date	30 October 2022
Student Name	R.Vigneshwari
Student Roll Number	820419106070
Maximum Marks	2 Marks

1. Download the dataset

2. Load the dataset into the tool.

In [1]:
```
import pandas as pd
data = pd.read_csv("Mall_Customers.csv")
data.head()
```

Out[1]:

	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 [2]:
```
data.shape
```

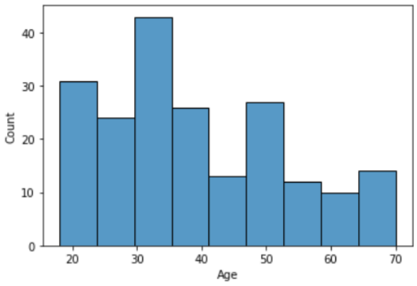
Out[2]: (200, 5)

3. Perform Below Visualizations

· Univariate Analysis

In [3]:
```
import seaborn as sns
sns.histplot(data, x="Age")
```

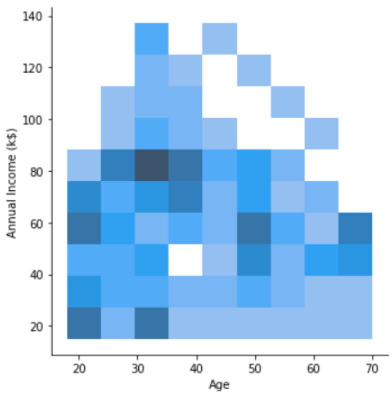
Out[3]: <AxesSubplot:xlabel='Age', ylabel='Count'>



In [ ]:
```
· Bi- Variate Analysis
```

In [4]:
```
sns.displot(data, x="Age", y="Annual Income (k$)")
```

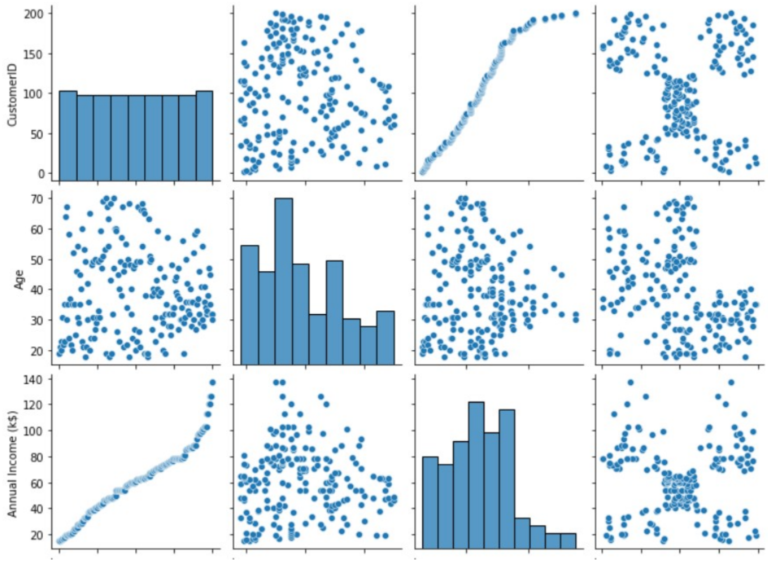
Out[4]: <seaborn.axisgrid.FacetGrid at 0x1c948fa0370>

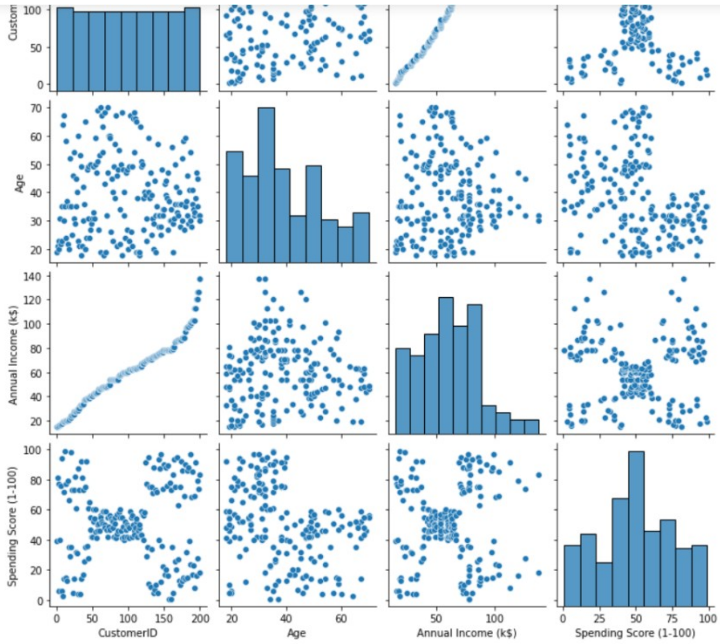


• Multi-Variate Analysis

In [6]:
```
sns.pairplot(data)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x1c94373c6d0>





4. Perform descriptive statistics on the dataset

In [7]:
```
data.mean()
```

```
C:\Users\welcome\AppData\Local\Temp/ipykernel_13532/531903386.py:1: FutureWarning: Dropping of nuisance columns in DataFrame re
ductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns be
fore calling the reduction.
  data.mean()
```

Out[7]:
```
CustomerID                100.50
Age                        38.85
Annual Income (k$)         60.56
Spending Score (1-100)     50.20
dtype: float64
```

In [8]:
```
data.median()
```

```
C:\Users\welcome\AppData\Local\Temp/ipykernel_13532/4184645713.py:1: FutureWarning: Dropping of nuisance columns in DataFrame r
eductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  data.median()
```

Out[8]:
```
CustomerID                100.5
Age                        36.0
Annual Income (k$)         61.5
Spending Score (1-100)     50.0
dtype: float64
```

In [9]:
```
data.mode()
```
Out[9]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Female	32.0	54.0	42.0
1	2	NaN	NaN	78.0	NaN
2	3	NaN	NaN	NaN	NaN
3	4	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN
...	...	...	...	...	...
195	196	NaN	NaN	NaN	NaN
196	197	NaN	NaN	NaN	NaN
197	198	NaN	NaN	NaN	NaN
198	199	NaN	NaN	NaN	NaN
199	200	NaN	NaN	NaN	NaN

200 rows × 5 columns

In [10]:
```
data.skew()
```
```
C:\Users\welcome\AppData\Local\Temp/ipykernel_13532/1188251951.py:1: FutureWarning: Dropping of nuisance columns in DataFrame r
eductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  data.skew()
```
Out[10]:
```
CustomerID                0.000000
Age                       0.485569
Annual Income (k$)        0.321843
Spending Score (1-100)   -0.047220
dtype: float64
```

In [11]:
```
data.kurt()
```
```
C:\Users\welcome\AppData\Local\Temp/ipykernel_13532/2907027414.py:1: FutureWarning: Dropping of nuisance columns in DataFrame r
eductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  data.kurt()
```
Out[11]:
```
CustomerID               -1.200000
Age                      -0.671573
Annual Income (k$)       -0.098487
Spending Score (1-100)   -0.826629
dtype: float64
```

In [12]:
```
data.std()
```
```
C:\Users\welcome\AppData\Local\Temp/ipykernel_13532/2723740006.py:1: FutureWarning: Dropping of nuisance columns in DataFrame r
eductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  data.std()
```
Out[12]:
```
CustomerID                57.879185
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

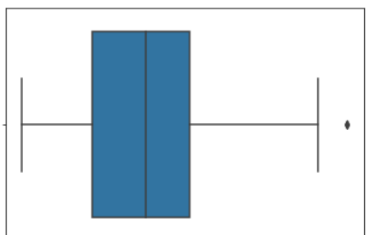
In [13]:
```
data.var()
```
```
C:\Users\welcome\AppData\Local\Temp/ipykernel_13532/445316826.py:1: FutureWarning: Dropping of nuisance columns in DataFrame re
ductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns be
fore calling the reduction.
  data.var()
```
Out[13]:
```
CustomerID                3350.000000
Age                        195.133166
Annual Income (k$)         689.835578
Spending Score (1-100)     666.854271
dtype: float64
```

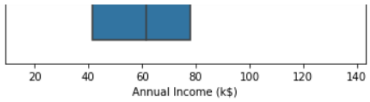
5. Check for Missing values and deal with them

In [14]:
```
data.isna().sum()
```
Out[14]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

6. Find the outliers and replace them outliers

In [15]:
```
sns.boxplot(data['Annual Income (k$)'])
```
```
C:\Users\welcome\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword
arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```
Out[15]:
```
<AxesSubplot:xlabel='Annual Income (k$)'>
```



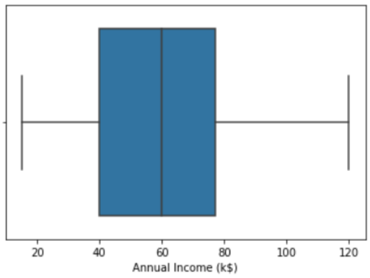


In [16]:
```
import numpy as np
data['Annual Income (k$)']=np.where(data['Annual Income (k$)']>120,20,data['Annual Income (k$)']) #replacing
```

In [17]:
```
sns.boxplot(data['Annual Income (k$)'])
```

```
C:\Users\welcome\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword
arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]:
```
<AxesSubplot:xlabel='Annual Income (k$)'>
```



7. Check for Categorical columns and perform encoding

In [18]:
```
data.head()
```

Out[18]:

	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 [19]:
```
from sklearn.preprocessing import LabelEncoder
```

In [20]:
```
le=LabelEncoder()
```

In [21]:
```
data['Gender']=le.fit_transform(data['Gender'])
data.head()
```

Out[21]:

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

8. Scaling the data

In [22]:
```
from sklearn.preprocessing import StandardScaler
```

In [23]:
```
scaler=StandardScaler()
x=scaler.fit_transform(data)
```

In [24]:
```
print(data)
```

```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0             1       1   19                  15                      39
1             2       1   21                  15                      81
2             3       0   20                  16                       6
3             4       0   23                  16                      77
4             5       0   31                  17                      40
..          ...     ...  ...                 ...                     ...
195         196       0   35                 120                      79
196         197       0   45                  20                      28
197         198       1   32                  20                      74
198         199       1   32                  20                      18
199         200       1   30                  20                      83

[200 rows x 5 columns]
```

9. Perform any of the clustering algorithms

In [25]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

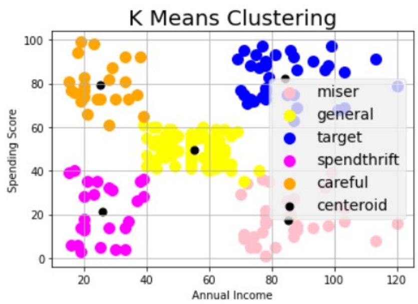
In [26]:
```
x = data.iloc[:, [3, 4]].values
```

In [27]:
```
print(x.shape)
```

```
(200, 2)
```

In [28]:
```
km = KMeans(n_clusters = 5, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
y_means = km.fit_predict(x)
```

In [29]:
```
plt.scatter(x[y_means == 0, 0], x[y_means == 0, 1], s = 100, c = 'pink', label = 'miser')
plt.scatter(x[y_means == 1, 0], x[y_means == 1, 1], s = 100, c = 'yellow', label = 'general')
plt.scatter(x[y_means == 2, 0], x[y_means == 2, 1], s = 100, c = 'blue', label = 'target')
plt.scatter(x[y_means == 3, 0], x[y_means == 3, 1], s = 100, c = 'magenta', label = 'spendthrift')
plt.scatter(x[y_means == 4, 0], x[y_means == 4, 1], s = 100, c = 'orange', label = 'careful')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:, 1], s = 50, c = 'black' , label = 'centeroid')
plt.style.use('fivethirtyeight')
plt.title('K Means Clustering', fontsize = 20)
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.grid()
plt.show()
```



10. Add the cluster data with the primary dataset

In [30]:
```
from sklearn.cluster import KMeans
```

In [31]:
```
km=KMeans(n_clusters=3, random_state=0)
```

In [32]:
```
data['Group or Cluster'] = km.fit_predict(data)
```

In [34]:
```
data.head()
```

Out[34]:

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

11. Split the data into dependent and independent variables.

In [35]:
```
y=data['Spending Score (1-100)']
y.head()
```

Out[35]:
```
0    39
1    81
2     6
3    77
4    40
Name: Spending Score (1-100), dtype: int64
```

In [36]:
```
x=data.drop(columns=['Spending Score (1-100)'])
x.head()
```

Out[36]:

	CustomerID	Gender	Age	Annual Income (k$)	Group or Cluster
0	1	1	19	15	2
1	2	1	21	15	2
2	3	0	20	16	2
3	4	0	23	16	2
4	5	0	31	17	2

12. Split the data into training and testing

In [37]:
```
from sklearn.model_selection import train_test_split
```

In [38]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [39]:
```
print(x_train.shape,y_train.shape)
```

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

In [40]:
```
print(x_test.shape,y_test.shape)
```

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

13. Build the Model

In [41]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

14. Train the Model

In [42]:
```
model.fit(x_train,y_train)
```

Out[42]:
```
DecisionTreeClassifier()
```

15. Test the Model

In [43]:
```
pred2=model.predict(x_test)

pred2
```

Out[43]:
```
array([49, 13,  4, 90,  5, 54, 42, 95, 35, 81, 42, 16, 27, 55, 35, 89, 45,
       16, 52, 55, 40, 99, 72, 42, 55, 46, 55, 55, 71, 40, 99, 15, 60, 55,
       16, 55, 40, 71, 16, 42], dtype=int64)
```

16. Measure the performance using Evaluation Metrics.

In [44]:
```
from sklearn import metrics
```

In [45]:
```
metrics.confusion_matrix(y_test,pred2)
```

Out[45]:
```
array([[0, 0, 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]], dtype=int64)
```

In [46]:
```
print('DT model ACcuracy Score:',metrics.accuracy_score(y_test,pred2))
```

```
DT model ACcuracy Score: 0.025
```


In [45]:
```
metrics.confusion_matrix(y_test,pred2)
```

Out[45]:
```
array([[0, 0, 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]], dtype=int64)
```

In [46]:
```
print('DT model ACcuracy Score:',metrics.accuracy_score(y_test,pred2))
```

```
DT model ACcuracy Score: 0.025
```


In [47]:
```
acc=metrics.accuracy_score(y_test,pred2)
acc
```

Out[47]:
```
0.025
```

In [48]:
```
1-acc
```

Out[48]:
```
0.975
```

In [49]:
```
data.head()
```

Out[49]:

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