In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```python
df=pd.read_csv("Mall_Customers.csv")
```

In [3]:
```python
df.head()
```

Out[3]:

	CustomerID	Genre	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 [4]:
```python
df.shape
```

Out[4]:
```
(200, 5)
```

In [5]:
```python
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   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
```

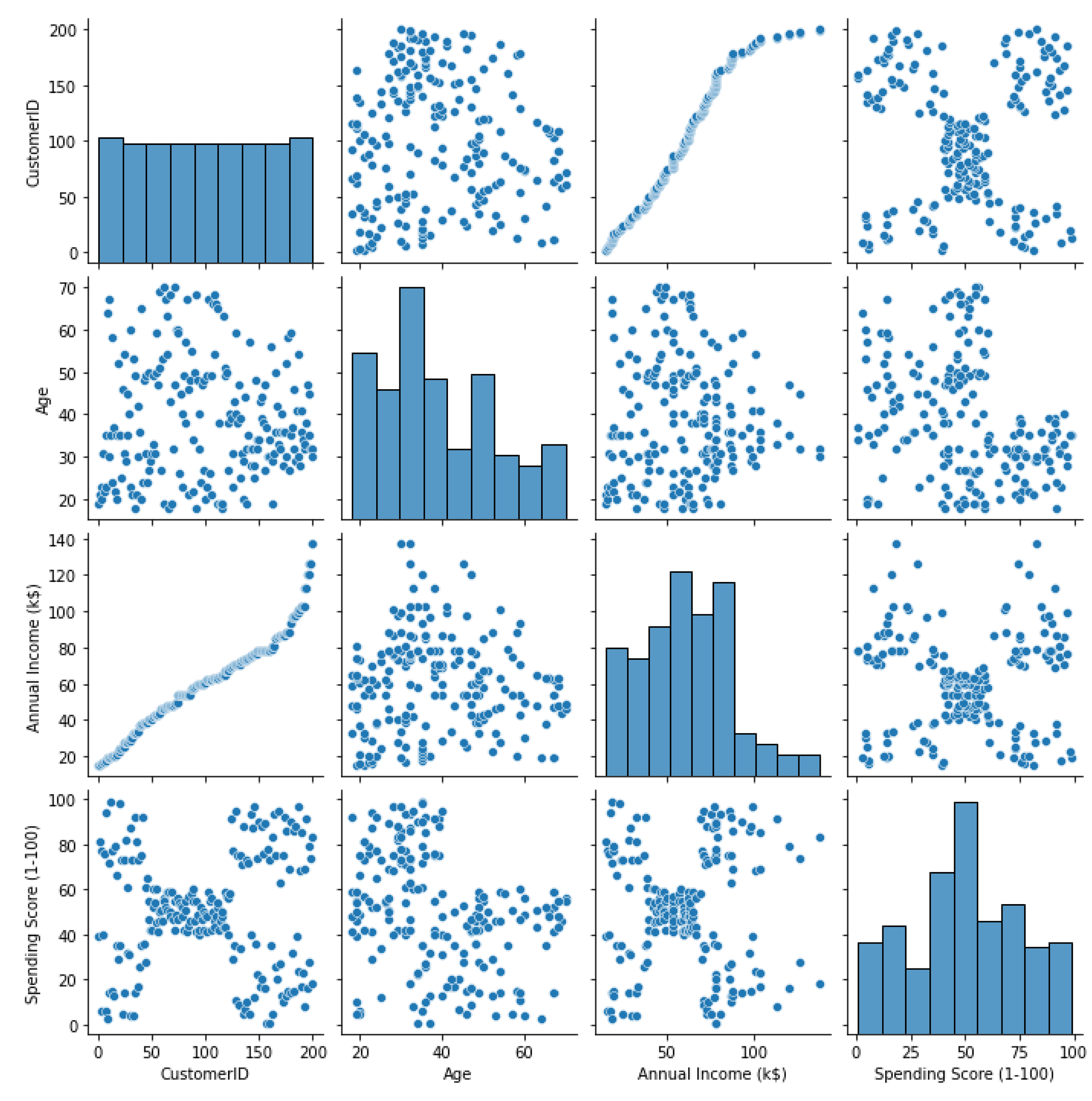
In [6]:
```python
df.describe()
```

Out[6]:

	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 [7]:
```python
sns.pairplot(df)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1fd30cb6ac0>
```



In [8]:
```python
x=df.iloc[:,[3,4]]
x
```

Out[8]:

	Annual Income (k$)	Spending Score (1-100)
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

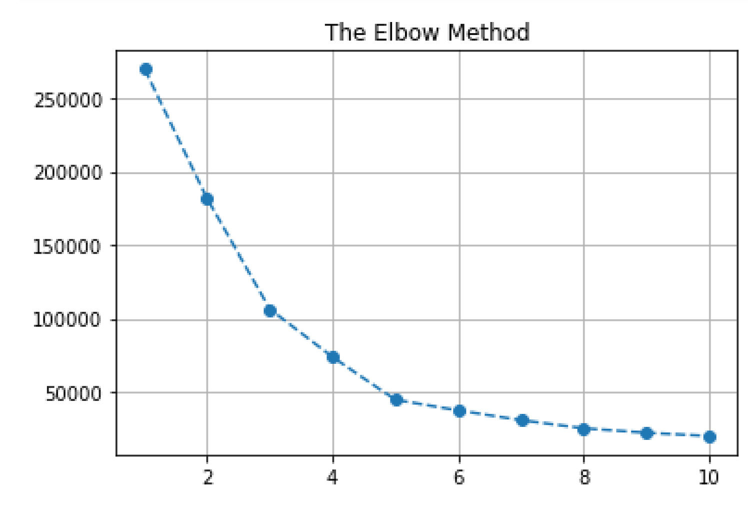
In [9]:
```python
from sklearn.cluster import KMeans
wcss = []

for i in range(1,11):
    Kmeans=KMeans(n_clusters=i,random_state=1)
    Kmeans.fit(x)
    wcss.append(Kmeans.inertia_)
```

In [10]:
```python
wcss
```

Out[10]:
```
[269981.28,
 181363.59595959593,
 106348.37306211118,
 73679.78903948836,
 44448.45544793371,
 37233.81451071001,
 30566.45113025186,
 25005.55037243283,
 21996.523372372307,
 19746.911957660894]
```

In [11]:
```python
plt.plot(range(1,11),wcss,"o--")
plt.title("The Elbow Method")
plt.grid()
plt.show()
```



In [12]:
```python
Kmeans=KMeans(n_clusters=5,random_state=1)
ylabel=Kmeans.fit_predict(x)
```

In [13]:
```python
df
```

Out[13]:

	CustomerID	Genre	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
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

In [14]:
```python
df["yKmeans"]=ylabel
df
```

Out[14]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)	yKmeans
0	1	Male	19	15	39	4
1	2	Male	21	15	81	0
2	3	Female	20	16	6	4
3	4	Female	23	16	77	0
4	5	Female	31	17	40	4
...	...	...	...	...	...	...
195	196	Female	35	120	79	3
196	197	Female	45	126	28	1
197	198	Male	32	126	74	3
198	199	Male	32	137	18	1
199	200	Male	30	137	83	3

200 rows × 6 columns

In [15]:
```python
Kmeans.cluster_centers_
```

Out[15]:
```
array([[25.72727273, 79.36363636],
       [88.2       , 17.11428571],
       [55.2962963 , 49.51851852],
       [86.53846154, 82.12820513],
       [26.30434783, 20.91304348]])
```

In [16]:
```python
df["yKmeans"].value_counts()
```

Out[16]:
```
2    81
3    39
1    35
4    23
0    22
Name: yKmeans, dtype: int64
```

In [17]:
```python
y=df.iloc[:,-1]
y
```

Out[17]:
```
0      4
1      0
2      4
3      0
4      4
      ..
195    3
196    1
197    3
198    1
199    3
Name: yKmeans, Length: 200, dtype: int32
```

In [18]:
```python
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1)
```

In [19]:
```python
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
```

In [20]:
```python
def indmodel(model):
    model.fit(xtrain,ytrain)
    ypred=model.predict(xtest)

    train=model.score(xtrain,ytrain)
    test=model.score(xtest,ytest)

    print(f' Training Accuracy : {train} \n Testing Accuracy : {test} \n')
    print(classification_report(ytest,ypred))

    return model
```

In [21]:
```python
gnb=indmodel(GaussianNB())
```

```
 Training Accuracy : 0.9785714285714285
 Testing Accuracy : 0.9833333333333333

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00        11
           2       0.95      1.00      0.98        21
           3       1.00      1.00      1.00        11
           4       1.00      0.89      0.94         9

    accuracy                           0.98        60
   macro avg       0.99      0.98      0.98        60
weighted avg       0.98      0.98      0.98        60
```

In [22]:
```python
knn=indmodel(KNeighborsClassifier(n_neighbors=5))
```

```
 Training Accuracy : 0.9714285714285714
 Testing Accuracy : 0.9833333333333333

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00        11
           2       0.95      1.00      0.98        21
           3       1.00      1.00      1.00        11
           4       1.00      0.89      0.94         9

    accuracy                           0.98        60
   macro avg       0.99      0.98      0.98        60
weighted avg       0.98      0.98      0.98        60
```

In [23]:
```python
logreg=indmodel(LogisticRegression())
```

```
 Training Accuracy : 0.9928571428571429
 Testing Accuracy : 0.95

              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      0.91      0.95        11
           2       0.88      1.00      0.93        21
           3       1.00      1.00      1.00        11
           4       1.00      0.89      0.94         9

    accuracy                           0.95        60
   macro avg       0.97      0.93      0.95        60
weighted avg       0.96      0.95      0.95        60
```

In [24]:
```python
cladt=indmodel(DecisionTreeClassifier(random_state=2))
```

```
 Training Accuracy : 1.0
 Testing Accuracy : 0.9666666666666667

              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      1.00      1.00        11
           2       0.91      1.00      0.95        21
           3       1.00      1.00      1.00        11
           4       1.00      0.89      0.94         9

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

In [ ]: