

TEAM ID	PNT2022TMID17480
STUDENT NAME	Lavanya.s
DOMAIN NAME	Artificial intelligence
PROJECT NAME	Early detection of chronic kidney disease using machine learning
MAXIMUM MARKS	2 MARKS

```
✓ [20] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. LOAD DATASET

2.load dataset

```
✓ [21] file=pd.read_csv("/content/Mall_Customers.csv")
0s df=pd.DataFrame(file)
df.head()
```

	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

```
✓ df['Gender']=df['Gender'].astype ('category')
```

```
✓ [13] df.head()
```

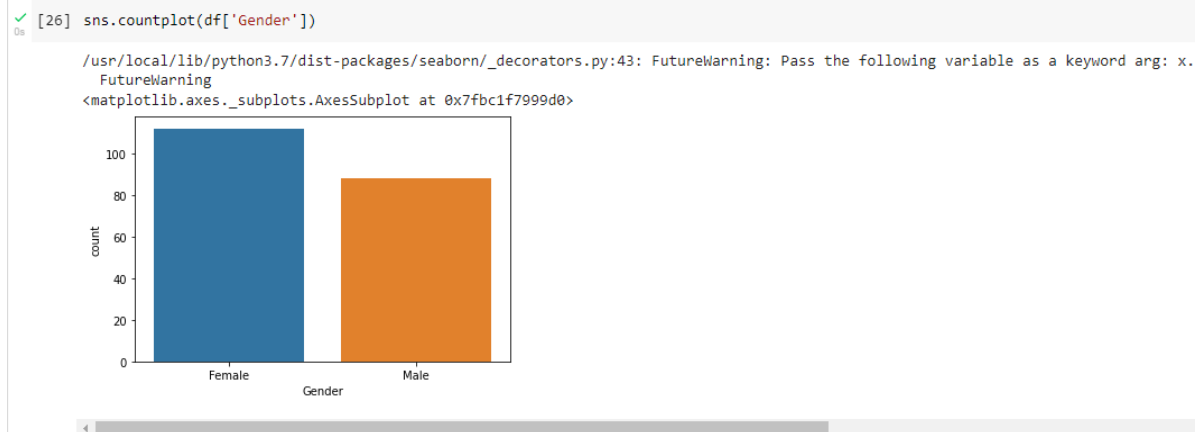
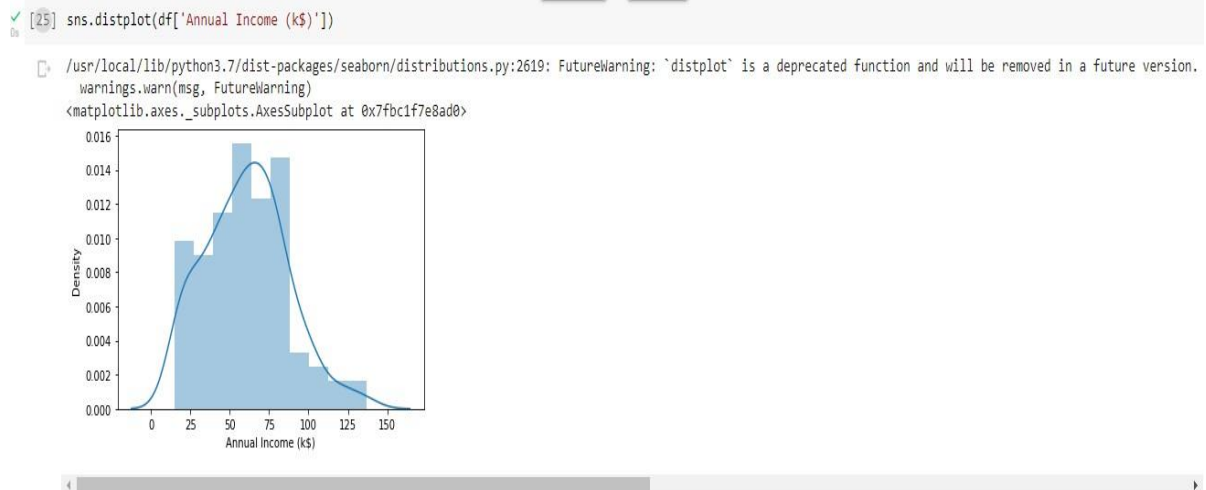
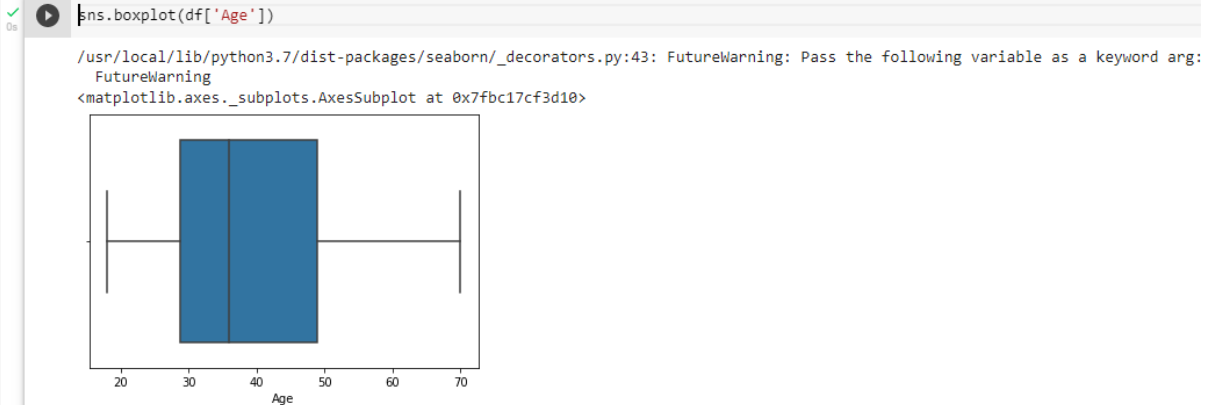
	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

3. PERFORM BELOW VISUALIZATIONS

• UNIVARIATE ANALYSIS

3.Perform Below Visualizations.

univariate analysis

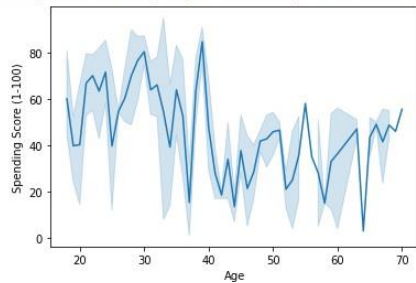


• BI- VARIATE ANALYSIS

bivariate analysis

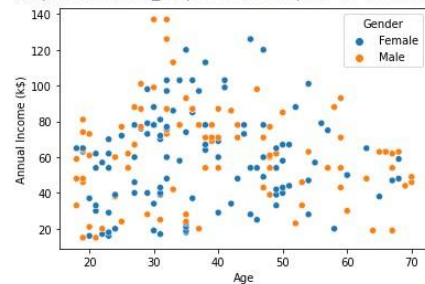
```
✓ [68] sns.lineplot(df['Age'],df['Spending Score (1-100)'])  
1s
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. I
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbc17da8f50>



```
✓ [28] sns.scatterplot(df['Age'],df['Annual Income (k$)'],hue=df["Gender"])  
1s
```

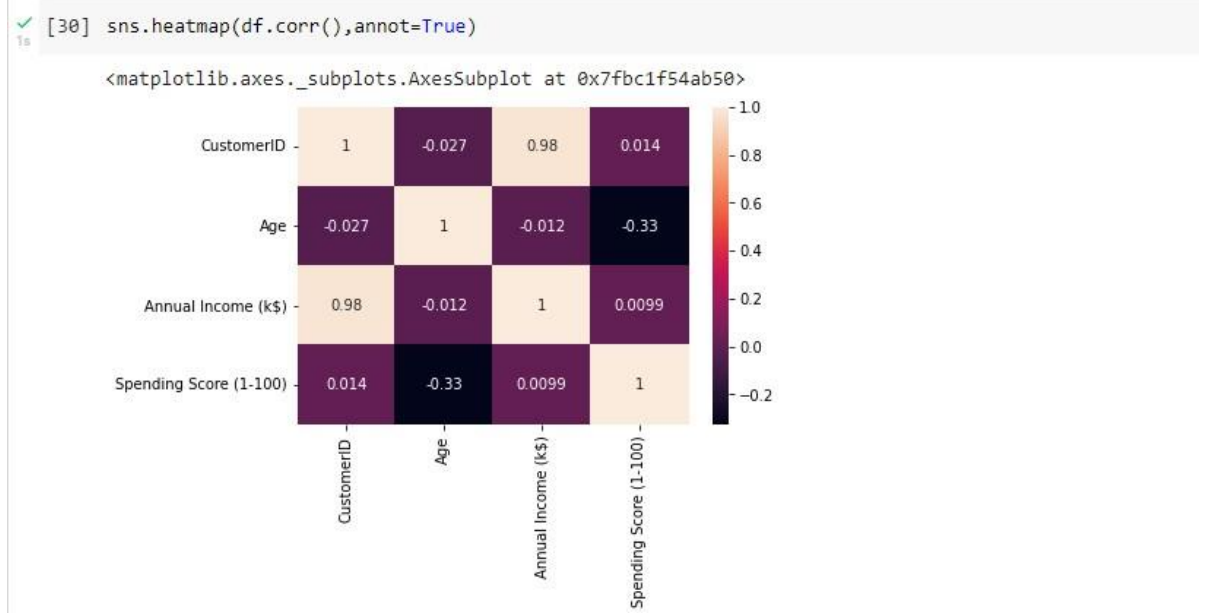
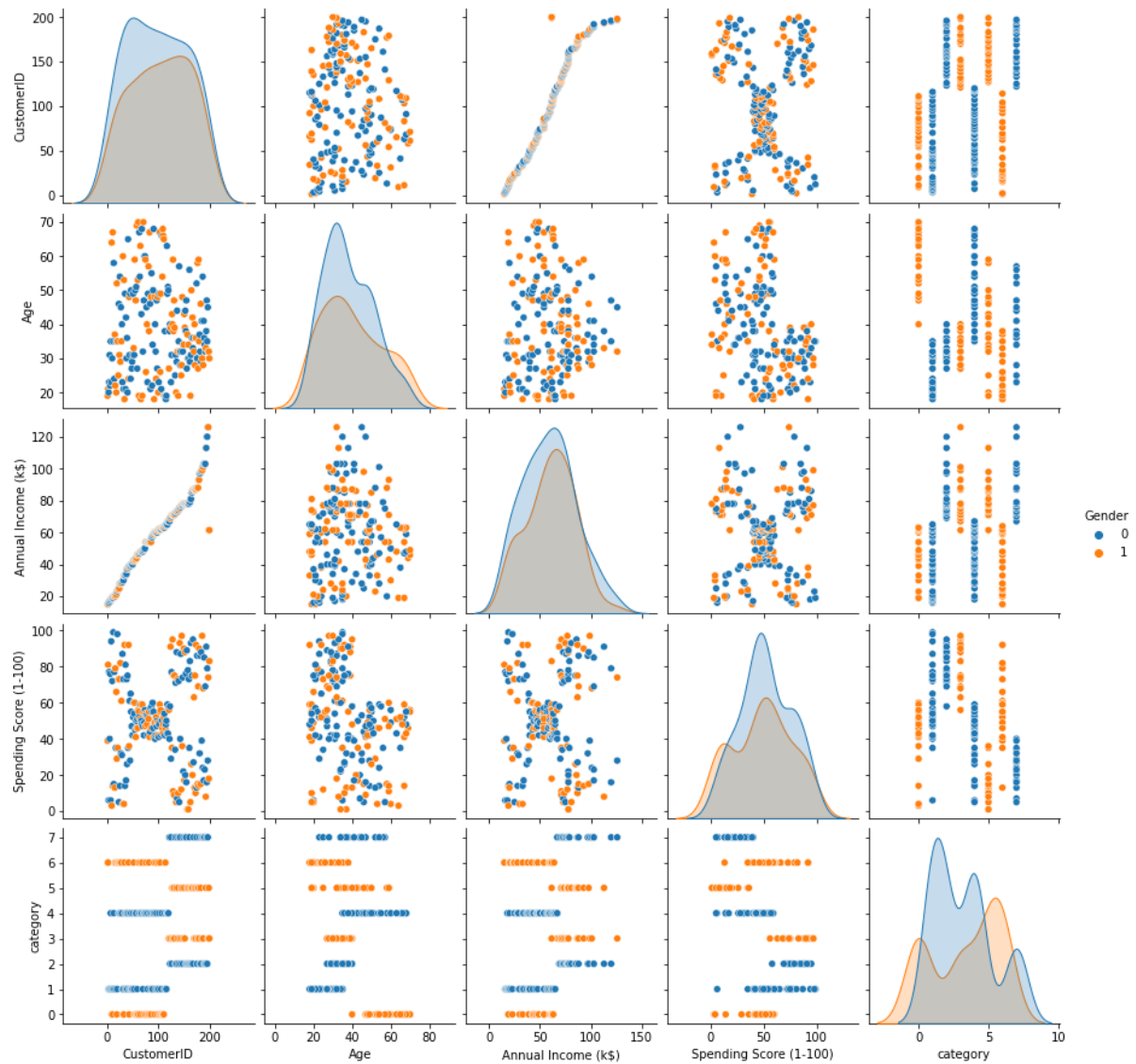
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y.
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbc1f6a2890>



• MULTI-VARIATE ANALYSIS

```
✓ [14s] sns.pairplot(df,hue='Gender')
```

<seaborn.axisgrid.PairGrid at 0x7fbc17cc9d10>



4. PERFORM DESCRIPTIVE STATISTICS ON THE DATASET

4.Perform descriptive statistics on the dataset.

```
✓ [71] df.describe()
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)	category
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	0.440000	38.850000	59.805000	50.200000	3.270000
std	57.879185	0.497633	13.969007	25.110699	25.823522	2.247746
min	1.000000	0.000000	18.000000	15.000000	1.000000	0.000000
25%	50.750000	0.000000	28.750000	41.500000	34.750000	1.000000
50%	100.500000	0.000000	36.000000	61.250000	50.000000	3.000000
75%	150.250000	1.000000	49.000000	77.250000	73.000000	5.000000
max	200.000000	1.000000	70.000000	126.000000	99.000000	7.000000

5. CHECK FOR MISSING VALUES AND DEAL WITH THEM

5.Check for Missing values and deal with them.

```
✓ [72] df.isnull().sum()
```

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

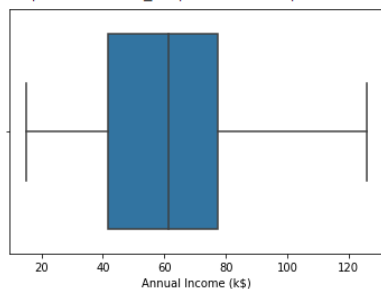
NO NULL VALUES

6. FIND THE OUTLIERS AND REPLACE THEM OUTLIERS

6.Find the outliers and replace them outliers

```
✓ [82] sns.boxplot(df['Annual Income (k$)'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x.
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbc18e64050>
```



```
✓ [34] #iqr median replacement
0s q1=df['Annual Income (k$)'].quantile(0.25)
q3=df['Annual Income (k$)'].quantile(0.75)
iqr=q3-q1
iqr
```

36.5

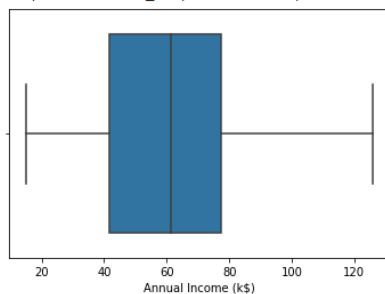
```
✓ [35] upperlimit=q3+1.5*iqr
0s lowerlimit=q1-1.5*iqr
print(upperlimit,lowerlimit)
```

132.75 -13.25

```
✓ [36] df["Annual Income (k$)"]=np.where(df["Annual Income (k$)"]>upperlimit,df['Annual Income (k$)'].median(),df["Annual Income (k$)"])
```

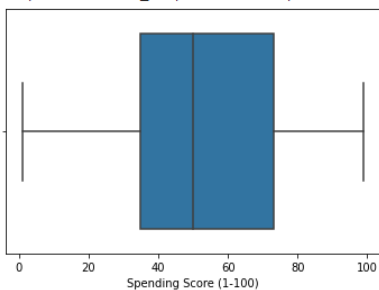
```
✓ [37] sns.boxplot(df['Annual Income (k$)'])
0s
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x.
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbc1c613610>



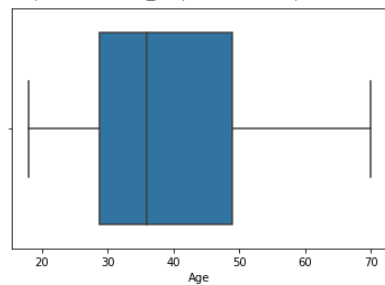
```
✓ [38] sns.boxplot(df['Spending Score (1-100)'])
0s
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x.
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbc1c5c3650>



```
✓ [41] sns.boxplot(df['Age'])
0s
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x.
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbc1ab2c190>



```
✓ [40] df.shape
0s
```

(200, 5)

7. CHECK FOR CATEGORICAL COLUMNS AND PERFORM ENCODING

7.Check for Categorical columns and perform encoding.

```
✓ [73] from sklearn.preprocessing import LabelEncoder  
0s le=LabelEncoder()  
df['Gender']=le.fit_transform(df['Gender'])  
df.head() # male-1 female-0
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)	category
0	1	1	19	15.0	39	6
1	2	1	21	15.0	81	6
2	3	0	20	16.0	6	1
3	4	0	23	16.0	77	1
4	5	0	31	17.0	40	1

8. SCALING THE DATA

8.Scaling the data

```
✓ [74] from sklearn.preprocessing import StandardScaler  
0s sc=StandardScaler()  
df1=sc.fit_transform(df)  
df1
```

```
array([[ -1.7234121,  1.12815215, -1.42456879, -1.78877673, -0.43480148,  
         1.21759788],  
       [-1.70609137,  1.12815215, -1.28103541, -1.78877673,  1.19570407,  
         1.21759788],  
       [-1.68877065, -0.88640526, -1.3528021 , -1.74885313, -1.71591298,  
        -1.01243487],  
       ...,  
       [ 1.68877065,  1.12815215, -0.49160182,  2.64274245,  0.92395314,  
        -0.12042177],  
       [ 1.70609137,  1.12815215, -0.49160182,  0.0676705 , -1.25005425,  
         0.77159133],  
       [ 1.7234121 ,  1.12815215, -0.6351352 ,  0.0676705 ,  1.27334719,  
        -0.12042177]])
```

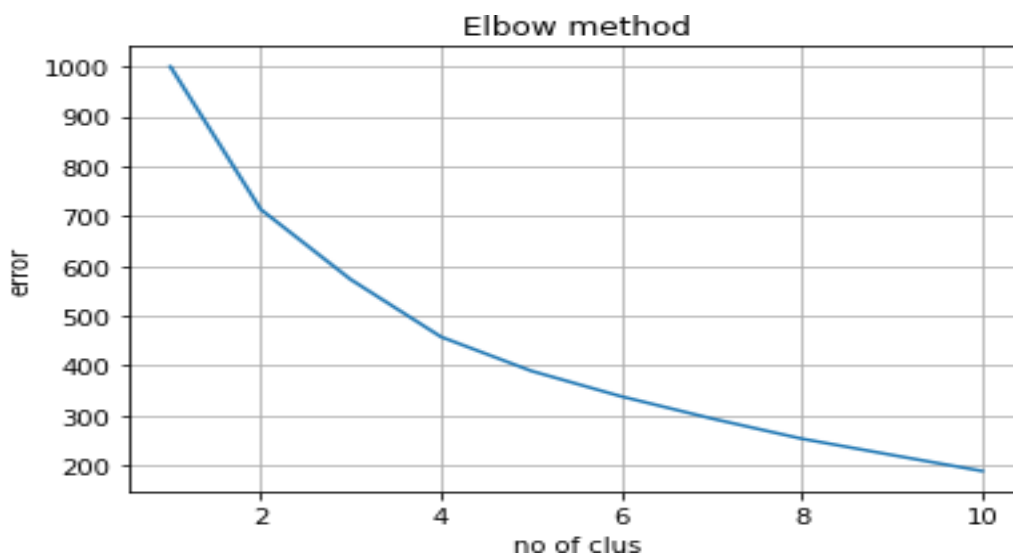
```
✓ [44] df1.shape  
0s (200, 5)
```


9. PERFORM ANY OF THE CLUSTERING ALGORITHMS

9.Perform any of the clustering algorithms

```
✓ [75] from sklearn.cluster import KMeans
0s
error=[]
for k in range(1,11):
    kmeans=KMeans(n_clusters=k,init='k-means++')
    kmeans.fit(df1)
    error.append(kmeans.inertia_)
```

```
✓ [46] import matplotlib.pyplot as plt
1s
plt.plot(range(1,11),error)
plt.title('Elbow method')
plt.xlabel('no of clus')
plt.ylabel('error')
plt.grid()
plt.show()
```



```
✓ [47] km=KMeans(n_clusters=8)
0s
category=km.fit_predict(df1)
category

array([6, 6, 1, 1, 1, 1, 4, 1, 0, 1, 0, 1, 4, 1, 6, 6, 1, 6, 0, 1, 6, 6,
       4, 6, 4, 6, 4, 6, 4, 1, 0, 1, 0, 6, 4, 1, 4, 1, 4, 1, 4, 6, 0, 1,
       4, 1, 4, 1, 1, 1, 4, 6, 1, 0, 4, 0, 4, 0, 1, 0, 0, 6, 4, 4, 0, 6,
       4, 4, 6, 1, 0, 4, 4, 4, 0, 6, 4, 0, 1, 4, 0, 6, 0, 4, 1, 0, 4, 1,
       1, 4, 4, 6, 0, 4, 1, 6, 4, 1, 0, 6, 1, 4, 0, 6, 0, 1, 4, 0, 0, 0,
       0, 1, 4, 6, 1, 1, 4, 4, 4, 4, 3, 7, 2, 3, 7, 2, 5, 3, 5, 3, 5, 3,
       7, 2, 5, 2, 7, 3, 5, 2, 7, 3, 7, 2, 5, 3, 5, 2, 7, 3, 5, 3, 7, 2,
       7, 2, 5, 2, 5, 2, 7, 2, 5, 2, 5, 2, 5, 2, 7, 3, 5, 3, 5, 3, 7, 2,
       5, 3, 5, 3, 7, 2, 5, 2, 7, 3, 7, 3, 7, 2, 7, 2, 5, 2, 7, 2, 7, 3,
       5, 3], dtype=int32)
```

10. ADD THE CLUSTER DATA WITH THE PRIMARY DATASET

10.Add the cluster data with the primary dataset

```
✓ [76] df['category']=pd.Series(category)
0s df.head()
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)	category
0	1	1	19	15.0	39	6
1	2	1	21	15.0	81	6
2	3	0	20	16.0	6	1
3	4	0	23	16.0	77	1
4	5	0	31	17.0	40	1

```
✓ [49] df.shape
0s
(200, 6)
```

11. SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES.

11.Split the data into dependent and independent variables.

```
✓ [77] y=df.iloc[:,-1]
0s y
0      6
1      6
2      1
3      1
4      1
..
195    2
196    7
197    3
198    5
199    3
Name: category, Length: 200, dtype: int32
```

```
✓ [51] X=df.iloc[:, :-1]
0s X
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	1	19	15.0	39
1	2	1	21	15.0	81
2	3	0	20	16.0	6
3	4	0	23	16.0	77
4	5	0	31	17.0	40
...
195	196	0	35	120.0	79
196	197	0	45	126.0	28
197	198	1	32	126.0	74
198	199	1	32	61.5	18
199	200	1	30	61.5	83

200 rows × 5 columns

12. SPLIT THE DATA INTO TRAINING AND TESTING

12.Split the data into training and testing

```
✓ [78] from sklearn.model_selection import train_test_split
0s
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)

✓ [53] print(X_train.shape)
0s print(X_test.shape)

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

13. BUILD THE MODEL

13.Build the Model

```
✓ [79] from sklearn.ensemble import RandomForestClassifier
0s model=RandomForestClassifier()
```

14. TRAIN THE MODEL

14.Train the Model

```
✓ [80] model.fit(X_train,y_train)
0s
RandomForestClassifier()
```

15. TEST THE MODEL

15. Test the Model

```
✓ [81] y_pred=model.predict(X_test)
0s y_pred

array([0, 5, 0, 0, 3, 5, 1, 5, 4, 7, 6, 2, 7, 7, 0, 1, 6, 5, 1, 0, 2, 3,
       1, 2, 0, 3, 3, 3, 4, 4, 1, 0, 5, 1, 4, 2, 3, 1, 4, 4], dtype=int32)
```

16. MEASURE THE PERFORMANCE USING EVALUATION METRICS.

16. Measure the performance using Evaluation Metrics.

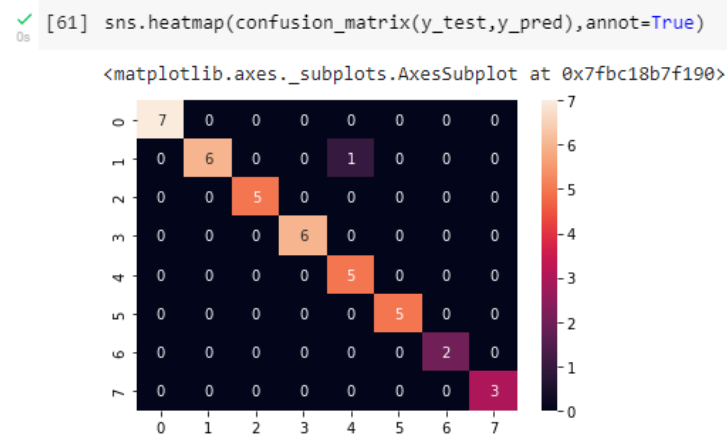
```
[57] from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

```
✓ [58] print('model accuracy', accuracy_score(y_test,y_pred))
0s model accuracy 0.975
```

```
✓ [59] train_pred=model.predict(X_train)
0s train_pred

array([5, 4, 4, 6, 7, 4, 0, 6, 4, 2, 4, 7, 7, 1, 3, 4, 0, 7, 4, 4, 3, 4,
       2, 1, 2, 0, 4, 2, 6, 4, 2, 4, 3, 4, 5, 4, 0, 6, 6, 3, 3, 4, 5, 2,
       0, 4, 1, 0, 7, 3, 1, 0, 7, 4, 7, 5, 2, 5, 4, 4, 5, 2, 4, 6, 1, 0,
       6, 0, 2, 6, 1, 1, 1, 0, 1, 2, 6, 7, 2, 6, 6, 1, 0, 1, 1, 4, 1, 4,
       5, 5, 4, 6, 1, 7, 2, 3, 5, 6, 3, 1, 4, 0, 1, 0, 6, 6, 3, 5, 0, 1,
       7, 3, 6, 4, 2, 0, 2, 5, 4, 4, 1, 3, 1, 5, 3, 5, 0, 7, 3, 2, 1, 6,
       0, 4, 1, 7, 2, 4, 0, 6, 2, 6, 7, 7, 7, 1, 1, 7, 1, 0, 1, 4, 6, 1,
       6, 4, 5, 4, 1, 5], dtype=int32)
```

```
✓ [60] print('model train accuracy',accuracy_score(y_train,train_pred))
0s model train accuracy 1.0
```



✓ [62] print(classification_report(y_test,y_pred))

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7
1	1.00	0.86	0.92	7
2	1.00	1.00	1.00	5
3	1.00	1.00	1.00	6
4	0.83	1.00	0.91	5
5	1.00	1.00	1.00	5
6	1.00	1.00	1.00	2
7	1.00	1.00	1.00	3
accuracy			0.97	40
macro avg	0.98	0.98	0.98	40
weighted avg	0.98	0.97	0.98	40