Assignment -4

Team ID	PNT2022TMID15174
Student Name	Charan C.D
Student Roll Number	7179KCTKCTKCTKCTKCTKCT19BEC130
Maximum Marks	2 Marks

Input

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
Matplotlib is building the font cache; this may take a moment.
```

```
df=pd.read_csv("Mall_Customers.csv")
df
```

Output

	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
...	...	...	...	...	...
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

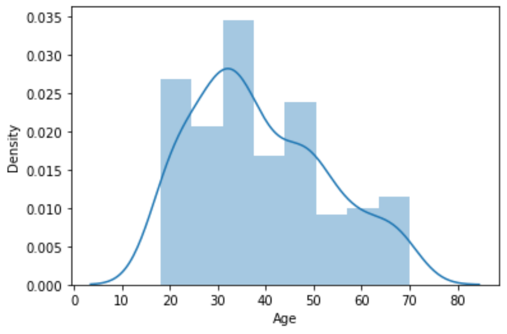
200 rows × 5 columns

Input

```
df.drop(["CustomerID"],axis="columns",inplace=True)
```

```
sns.distplot(df.Age)
```

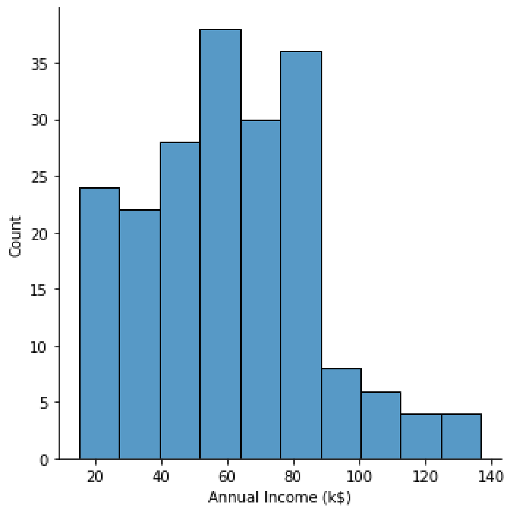
Output



Input

```
sns.displot(df["Annual Income (k$)"])
```

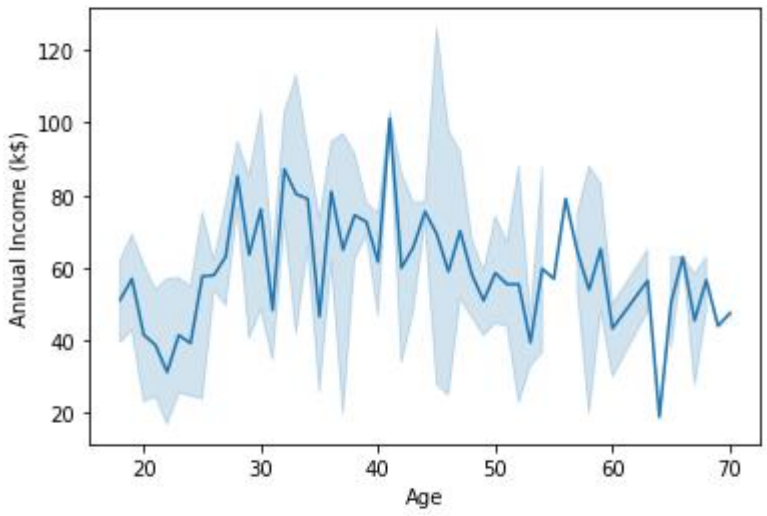
Output



Input

```
sns.lineplot(df.Age,df["Annual Income (k$)"])
```

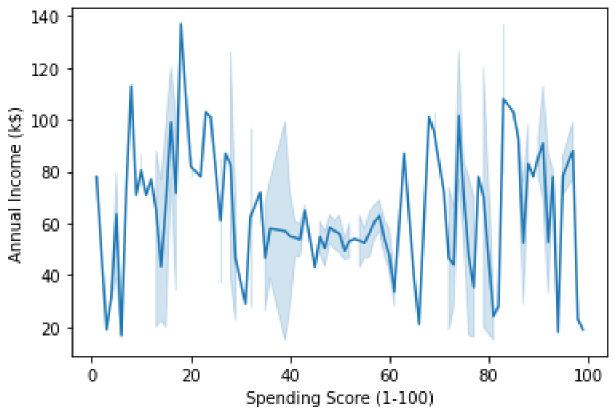
Output



Input

```
sns.lineplot(df["Spending Score (1-100)"],df["Annual Income (k$)"])
```

Output



Input

```
df.describe()
```

Output

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

Input

```
df.isnull().any()
```

Output

```
Gender                    False
Age                       False
Annual Income (k$)        False
Spending Score (1-100)    False
dtype: bool
```

Input

```
df.isnull().sum()
```

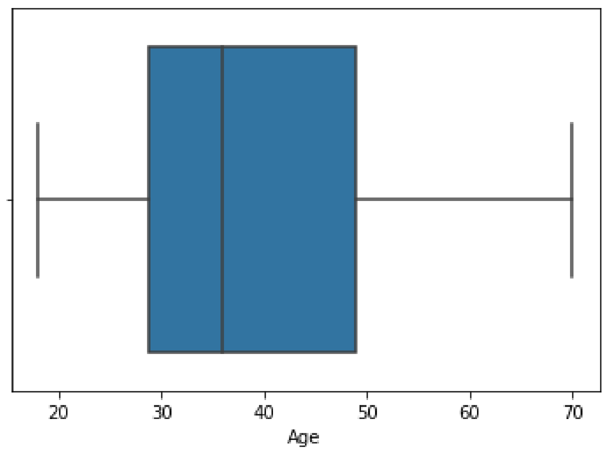
Output

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

Input

```
sns.boxplot(df.Age)
```

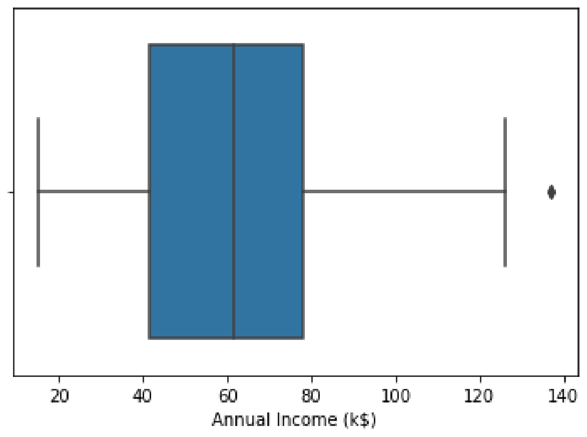
Output



Input

```
sns.boxplot(df["Annual Income (k$)"])
```

Output



Input

```
a=df["Annual Income (k$)"].quantile(0.99)
a
```

Output

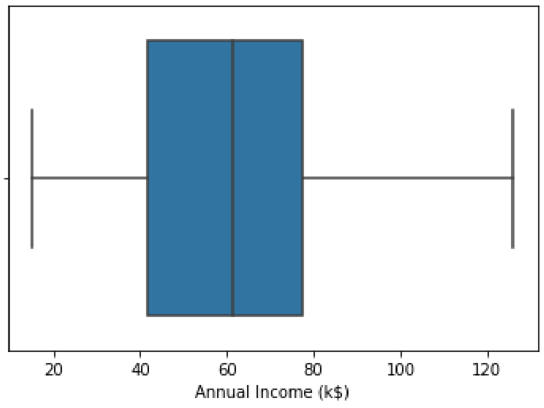
```
126.1099999999999
```

Input

```
df["Annual Income (k$)"]=np.where(df["Annual Income (k$)"]>=a,df["Annual Income
(k$)"].median(),df["Annual Income (k$)"])


sns.boxplot(df["Annual Income (k$)"])
```

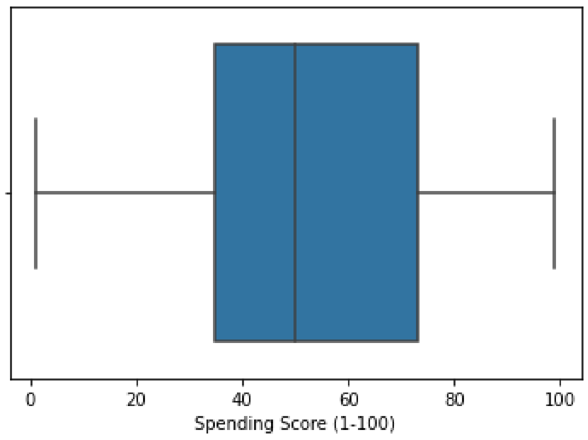
Output



Input

```
sns.boxplot(df["Spending Score (1-100)"])
```

Output

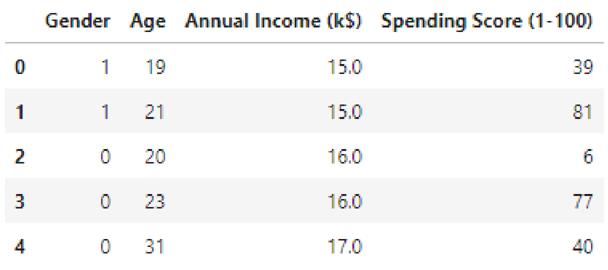


Input

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.Gender=le.fit_transform(df.Gender)

df.head()
```

Output



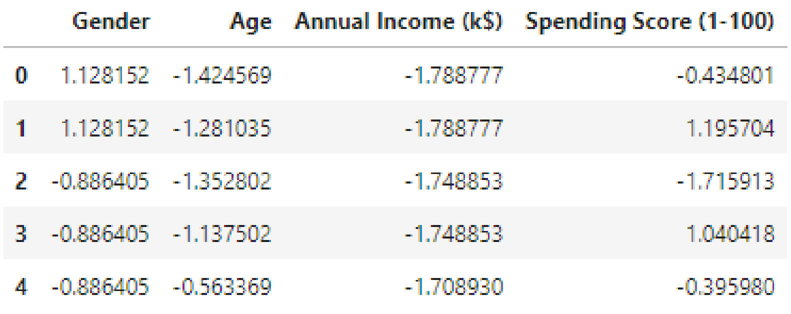
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15.0	39
1	1	21	15.0	81
2	0	20	16.0	6
3	0	23	16.0	77
4	0	31	17.0	40

Input

```
from sklearn.preprocessing import scale
df=pd.DataFrame(scale(df),columns=df.columns)


df.head()
```

Output



	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.128152	-1.424569	-1.788777	-0.434801
1	1.128152	-1.281035	-1.788777	1.195704
2	-0.886405	-1.352802	-1.748853	-1.715913
3	-0.886405	-1.137502	-1.748853	1.040418
4	-0.886405	-0.563369	-1.708930	-0.395980

Input

```
from sklearn.cluster import KMeans
model=KMeans(n_clusters=2)
model
```

Output

```
KMeans(n_clusters=2)
```

Input

```
y_predicted=model.fit_predict(df)
y_predicted
```

Output

```
array([0, 0, 1, 0, 0, 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, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 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, 0])
```

Input

```
df["clusters"]=y_predicted
df
```

Output

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	clusters
0	1.128152	-1.424569	-1.788777	-0.434801	0
1	1.128152	-1.281035	-1.788777	1.195704	0
2	-0.886405	-1.352802	-1.748853	-1.715913	1
3	-0.886405	-1.137502	-1.748853	1.040418	0
4	-0.886405	-0.563369	-1.708930	-0.395980	0
...	...	...	...	...	...
195	-0.886405	-0.276302	2.403201	1.118061	0
196	-0.886405	0.441365	2.642742	-0.861839	1
197	1.128152	-0.491602	2.642742	0.923953	0
198	1.128152	-0.491602	0.067670	-1.250054	1
199	1.128152	-0.635135	0.067670	1.273347	0

200 rows × 5 columns

Input

```
x=df.drop("clusters",axis="columns")
x
```

Output

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.128152	-1.424569	-1.788777	-0.434801
1	1.128152	-1.281035	-1.788777	1.195704
2	-0.886405	-1.352802	-1.748853	-1.715913
3	-0.886405	-1.137502	-1.748853	1.040418
4	-0.886405	-0.563369	-1.708930	-0.395980
...	...	...	...	...
195	-0.886405	-0.276302	2.403201	1.118061
196	-0.886405	0.441365	2.642742	-0.861839
197	1.128152	-0.491602	2.642742	0.923953
198	1.128152	-0.491602	0.067670	-1.250054
199	1.128152	-0.635135	0.067670	1.273347

200 rows × 4 columns

Input

```
y=df.clusters
y
```

Output

```
0      0
1      0
2      1
3      0
4      0
      ..
195    0
196    1
197    0
198    1
199    0
Name: clusters, Length: 200, dtype: int32
```

Input

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=23)


x_train.shape
```

Output

```
(160, 4)
```

Input

```
x_test.shape
```

Output

```
(40, 4)
```

Input

```
lr.fit(x_train,y_train)
```

Output

```
LogisticRegression()
```

Input

```
lr.score(x_test,y_test)
```

Output

```
1.0
```

Input

```
lr.score(x_train,y_train)
```

Output

```
1.0
```

Input

```
from sklearn.metrics import confusion_matrix
y_pred = lr.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

Output

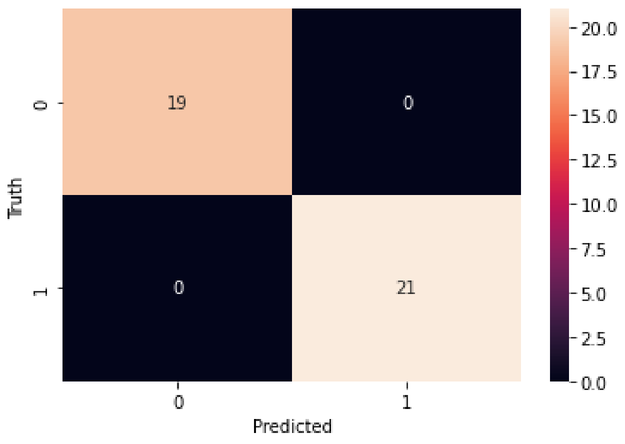
```
array([[19,  0],
       [ 0, 21]], dtype=int64)
```

Input

```
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Output

```
Text(33.0, 0.5, 'Truth')
```



Input

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Output

```
  precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        21

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```