```
import numpy as np
import pandas as pd
```

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

```
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

Next steps: Generate code with df   View recommended plots

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

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

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

```
X=df.drop(columns=['CustomerID'])
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
column_transformer=ColumnTransformer([
    ('onehot',OneHotEncoder(drop='first'),['Gender'])
],remainder='passthrough')
```

```
X_transformed=column_transformer.fit_transform(X)
```

```
X_transformed
```

```
        [  0.,  47., 120.,  16.],
        [  0.,  35., 120.,  79.],
        [  0.,  45., 126.,  28.],
        [  1.,  32., 126.,  74.],
        [  1.,  32., 137.,  18.],
        [  1.,  30., 137.,  83.]])
```

```
len(X_transformed)
```

```
200
```

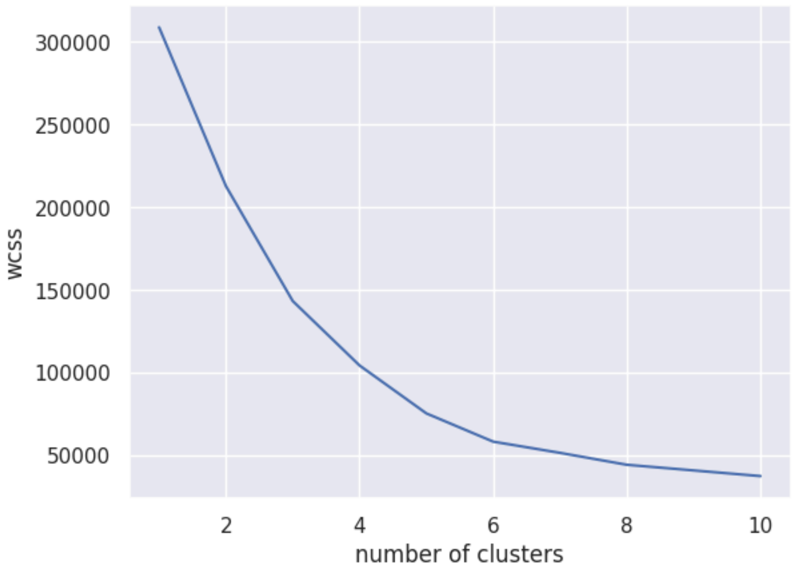
```
#Finding the correct number of Cluster using elbow method by finding wcss,we calculate wcss with taking k=1 and so on
wcss=[]
for i in range(1,11):
  kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
  kmeans.fit(X_transformed)
  wcss.append(kmeans.inertia_)#kmeans_.inertia gives wcss value
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
```

```
#plot the elbow graph
wcss
```

```
[308862.06000000006,
 212889.44245524303,
 143391.59236035676,
 104414.67534220168,
 75427.71182424155,
 58348.641363315044,
 51575.2779310779,
 44359.634641148325,
 40942.51117006117,
 37515.84125504126]
```

```
#we check the significant drop using the drop and take that cluster where it drops
sns.set()
plt.plot(range(1,11),wcss)
plt.xlabel('number of clusters')
plt.ylabel('wcss')
plt.show()
```



```
#By Looking at the graph we optimal number of clusters is 6 ,data points are highly packed when clusters are 6
algo=KMeans(n_clusters=6,init='k-means++',random_state=0)
#Now Lets Return a label for each cluster
Y=algo.fit_predict(X_transformed)
print(Y)
```

```
[5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5 4 5
 4 5 4 1 4 1 0 5 4 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 0 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 1 1 1
 0 0 0 0 0 1 1 1 1 0 0 0 3 0 3 2 3 2 3 2 3 0 3 2 3 2 3 2 3 2 3 0 3 2 3 2 3
 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2
 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3]
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
```

```
algo.cluster_centers_[0]
```

```
array([ 0.34210526, 27.        , 56.65789474, 49.13157895])
```

```
from sklearn.manifold import TSNE
data = pd.DataFrame(X_transformed)  # Assuming X_transformed is a NumPy array
data["cluster_label"] = Y
```

```
data
```

	0	1	2	3	cluster_label
0	1.0	19.0	15.0	39.0	5
1	1.0	21.0	15.0	81.0	4
2	0.0	20.0	16.0	6.0	5
3	0.0	23.0	16.0	77.0	4
4	0.0	31.0	17.0	40.0	5
...	...	...	...	...	...
195	0.0	35.0	120.0	79.0	3
196	0.0	45.0	126.0	28.0	2
197	1.0	32.0	126.0	74.0	3
198	1.0	32.0	137.0	18.0	2
199	1.0	30.0	137.0	83.0	3

200 rows × 5 columns

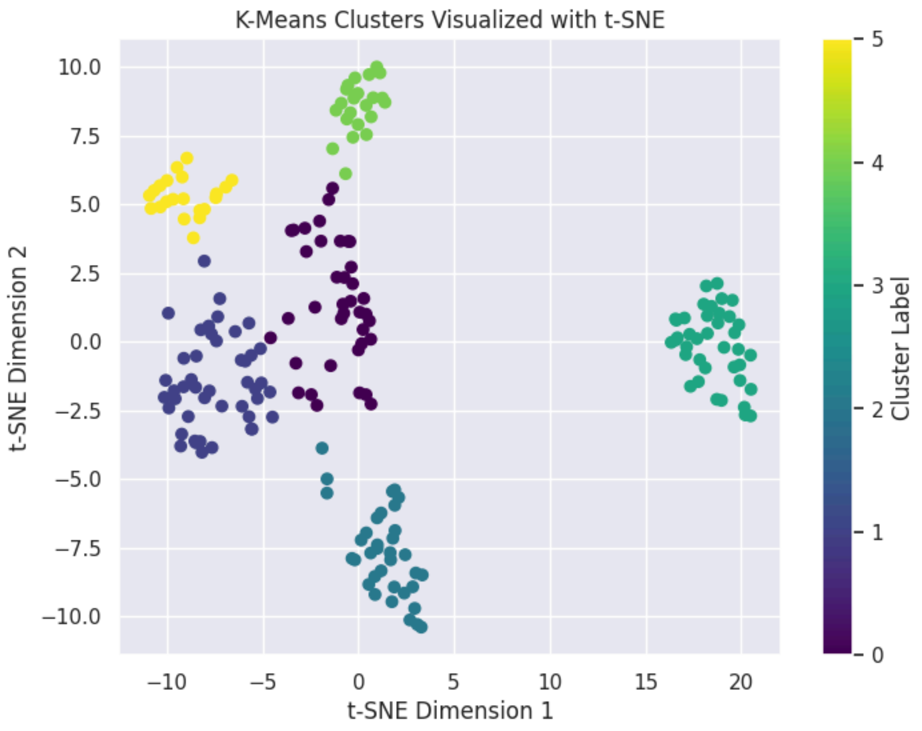
Next steps: Generate code with data   View recommended plots

```
tsne = TSNE(n_components=2, random_state=42)  # Set random_state for reproducibility
data_embedded = tsne.fit_transform(data.iloc[:, :-1])  # Exclude cluster label column
```

```
data_embedded
```

```
[ 2.05203457e+01, -1.73053467e+00],
[ 2.69720984e+00, -1.01214828e+01],
[ 2.01715431e+01, -2.38239288e+00],
[ 3.08672380e+00, -1.02859859e+01],
[ 2.02185097e+01, -2.66119075e+00],
[ 3.28934383e+00, -1.03831902e+01],
[ 2.05005169e+01, -2.69793367e+00]], dtype=float32)
```

```
plt.figure(figsize=(8, 6))
plt.scatter(data_embedded[:, 0], data_embedded[:, 1], c=data["cluster_label"], cmap="viridis")
plt.xlabel("t-SNE Dimension 1")
plt.ylabel("t-SNE Dimension 2")
plt.title("K-Means Clusters Visualized with t-SNE")
plt.colorbar(label="Cluster Label")
plt.show()
```



```
!pip install umap-learn
```

```
Collecting umap-learn
  Downloading umap_learn-0.5.6-py3-none-any.whl (85 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.7/85.7 kB 1.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.25.2)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.2.2)
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (0.58.1)
Collecting pynndescent>=0.5 (from umap-learn)
  Downloading pynndescent-0.5.12-py3-none-any.whl (56 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.8/56.8 kB 6.3 MB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from umap-learn) (4.66.4)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.51.2->umap-learn) (
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pynndescent>=0.5->umap-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->umap-learn) (3
Installing collected packages: pynndescent, umap-learn
Successfully installed pynndescent-0.5.12 umap-learn-0.5.6
```

```
from umap import UMAP
import plotly.express as px
features=X_transformed[:,:]
```

```
umap_2d=UMAP(n_components=2,init='random',random_state=0)
proj_2d=umap_2d.fit_transform(features)
```

```
/usr/local/lib/python3.10/dist-packages/umap/umap_.py:1945: UserWarning:

n_jobs value 1 overridden to 1 by setting random_state. Use no seed for parallelism.
```

```
proj_2d
```

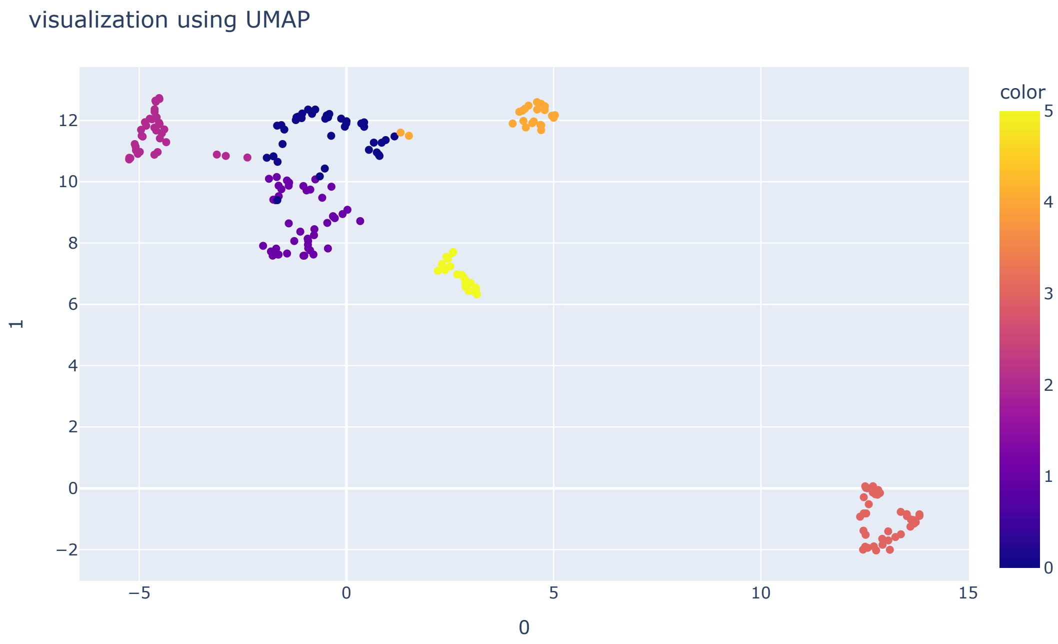
```
       [-9.62622464e-01,  9.71544170e+00],
       [-1.11286795e+00,  8.37064838e+00],
       [-5.21911383e-01,  1.04290905e+01],
       [-1.43240786e+00,  7.65775728e+00],
       [-1.03890216e+00,  9.85797882e+00],
       [ 1.95701304e-03,  1.19718199e+01],
       [-8.71639788e-01,  9.74322796e+00],
       [-1.39167261e+00,  8.64079666e+00],
       [-1.27637908e-01,  1.20574636e+01],
       [-3.68061513e-01,  1.14972610e+01],
       [-1.39249289e+00,  9.87040615e+00],
       [-1.65523767e+00,  7.62648535e+00],
       [-1.16842568e+00,  1.21205893e+01],
       [-1.38261771e+00,  9.97018909e+00],
       [-1.66118777e+00,  1.06482019e+01],
       [-1.54129660e+00,  1.12284870e+01],
       [-4.13181394e-01,  1.22099352e+01],
       [-1.43558073e+00,  1.00394926e+01],
       [-5.10951400e-01,  1.20548067e+01],
       [-1.68118203e+00,  1.01521225e+01],
       [-8.29786003e-01,  1.22136745e+01],
       [-1.22162068e+00,  1.20100889e+01],
       [-1.63629091e+00,  9.86967278e+00],
       [-1.63712645e+00,  7.62244511e+00],
       [-4.75261658e-01,  1.21626873e+01],
       [-1.63346064e+00,  9.52741528e+00],
       [-1.20340633e+00,  1.21015959e+01],
       [-1.82180274e+00,  7.72308779e+00],
       [-1.56842661e+00,  9.75531864e+00],
       [-1.69529068e+00,  7.81777143e+00],
       [-1.79117668e+00,  7.70102262e+00],
       [-1.78004003e+00,  7.58749628e+00],
       [-7.52228260e-01,  1.23531408e+01],
       [-1.76096892e+00,  1.08278170e+01],
       [-1.08143926e+00,  1.20724640e+01],
       [-1.06707680e+00,  1.22242107e+01],
       [-9.25371289e-01,  1.23534365e+01],
       [-2.01085711e+00,  7.90808725e+00],
```

```
plt.figure(figsize=(16,18))

fig_2d=px.scatter(
    proj_2d,x=0,y=1,labels=Y,color=Y
)
fig_2d.update_layout(
    title="visualization using UMAP",

)

fig_2d.show()
```



```
<Figure size 1600x1800 with 0 Axes>
```

Start coding or generate with AI.