```
import pandas as pd

import numpy as np

df=pd.read_csv('/content/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 | New interactive sheet

```
df.tail()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
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

```
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   Gender                  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
```

```
df.describe()
```

	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

```
df.shape
```

```
(200, 5)
```

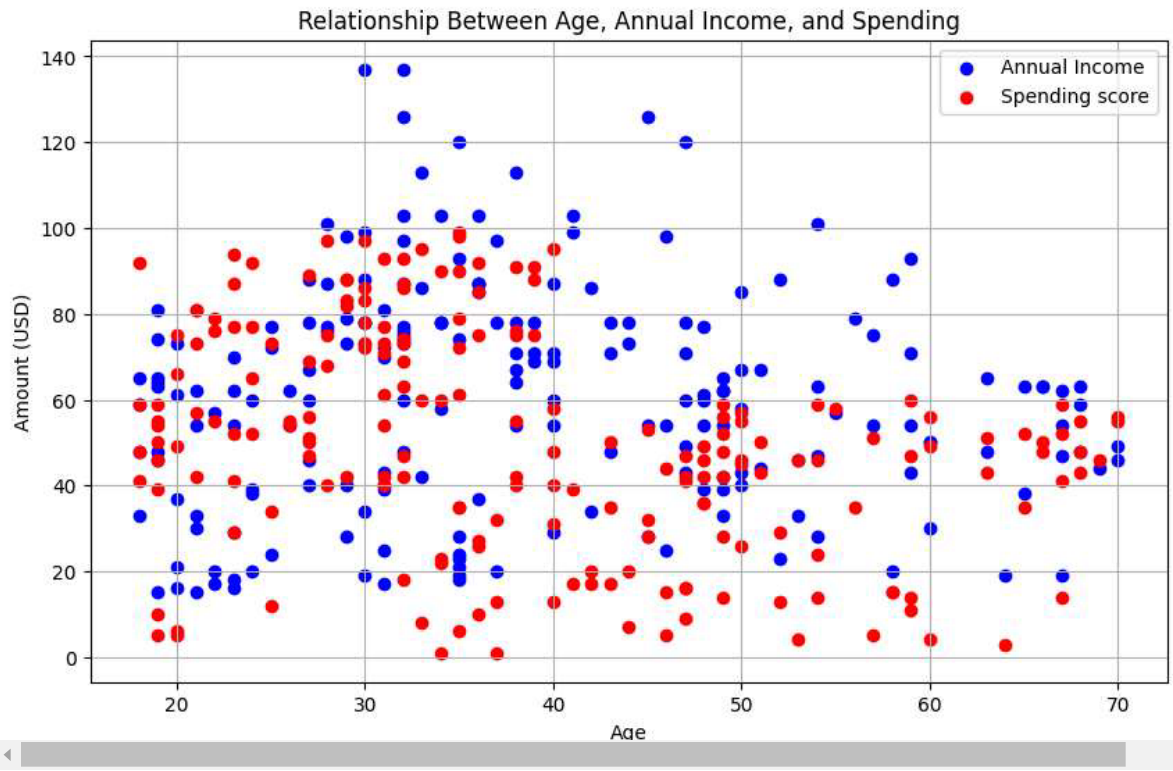
```
df.isnull().sum()
```

	0
CustomerID	0
Gender	0
Age	0
Annual Income (k$)	0
Spending Score (1-100)	0

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10, 6))
plt.scatter(df['Age'], df['Annual Income (k$)'], color='blue', label='Annual Income')
plt.scatter(df['Age'], df['Spending Score (1-100)'], color='red', label='Spending score')

plt.xlabel('Age')
plt.ylabel('Amount (USD)')
plt.title('Relationship Between Age, Annual Income, and Spending')
plt.legend()
plt.grid(True)
plt.show()
```




```
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 | New interactive sheet

```
features=['Annual Income (k$)','Spending Score (1-100)']

df=df[features].copy()

df.head()
```

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
from sklearn.preprocessing import StandardScaler

ss=StandardScaler()

ss_df=ss.fit_transform(df)

ss_df
```

```
[ 0.62750542,  1.81684904],
[ 0.62750542, -0.55126616],
[ 0.62750542,  0.92395314],
[ 0.66567484, -1.09476801],
[ 0.66567484,  1.54509812],
[ 0.66567484, -1.28887582],
[ 0.66567484,  1.46745499],
[ 0.66567484, -1.17241113],
[ 0.66567484,  1.00159627],
[ 0.66567484, -1.32769738],
[ 0.66567484,  1.50627656],
[ 0.66567484, -1.91002079],
[ 0.66567484,  1.07923939],
[ 0.66567484, -1.91002079],
[ 0.66567484,  0.88513158],
[ 0.70384427, -0.59008772],
[ 0.70384427,  1.27334719],
[ 0.78018313, -1.75473454],
[ 0.78018313,  1.6615628 ],
[ 0.93286085, -0.93948177],
[ 0.93286085,  0.96277471],
[ 0.97103028, -1.17241113],
[ 0.97103028,  1.73920592],
[ 1.00919971, -0.90066021],
[ 1.00919971,  0.49691598],
[ 1.00919971, -1.44416206],
[ 1.00919971,  0.96277471],
[ 1.00919971, -1.56062674],
[ 1.00919971,  1.62274124],
[ 1.04736914, -1.44416206],
[ 1.04736914,  1.38981187],
[ 1.04736914, -1.36651894],
[ 1.04736914,  0.72984534],
[ 1.23821628, -1.4053405 ],
[ 1.23821628,  1.54509812],
[ 1.390894  , -0.7065524 ],
[ 1.390894  ,  1.38981187],
[ 1.42906343, -1.36651894],
[ 1.42906343,  1.46745499],
[ 1.46723286, -0.43480148],
[ 1.46723286,  1.81684904],
[ 1.54357172, -1.01712489],
[ 1.54357172,  0.69102378],
[ 1.61991057, -1.28887582],
[ 1.61991057,  1.35099031],
[ 1.61991057, -1.05594645],
[ 1.61991057,  0.72984534],
[ 2.00160487, -1.63826986],
[ 2.00160487,  1.58391968],
[ 2.26879087, -1.32769738],
[ 2.26879087,  1.11806095],
[ 2.49780745, -0.86183865],
[ 2.49780745,  0.92395314],
[ 2.91767117, -1.25005425],
[ 2.91767117,  1.27334719]])
```

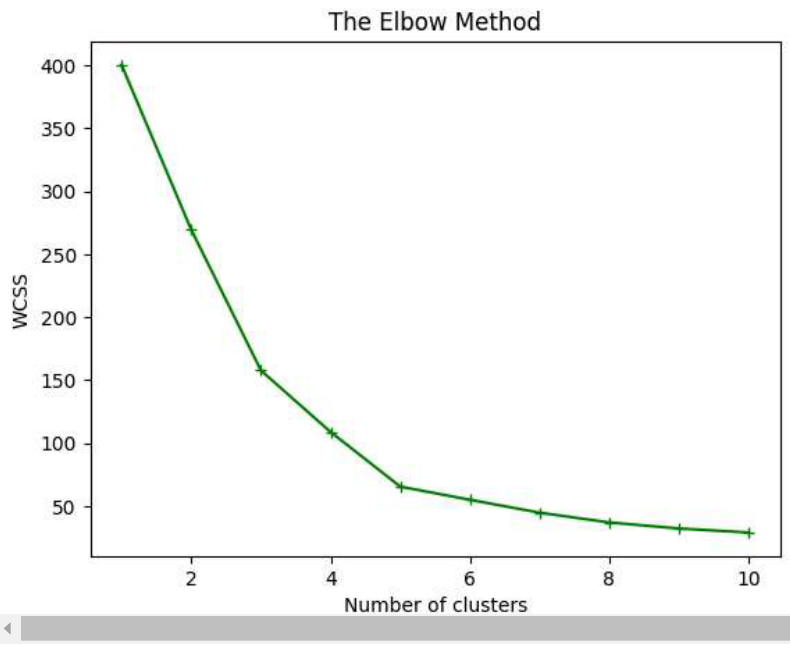
```
from sklearn.cluster import KMeans
```

```
wcss=[]

for i in range(1,11):
  kmeans=KMeans(n_clusters=i,init='k-means++',random_state=0)
  kmeans.fit(ss_df)
  wcss.append(kmeans.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
```

```
plt.plot(range(1,11),wcss,marker='+',color='green')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
km1=KMeans(n_clusters=5,init='k-means++',random_state=0)
```

```
km1.fit(ss_df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
```

KMeans

KMeans(n_clusters=5, random_state=0)

```
cent1=km1.cluster_centers_
cent1
```

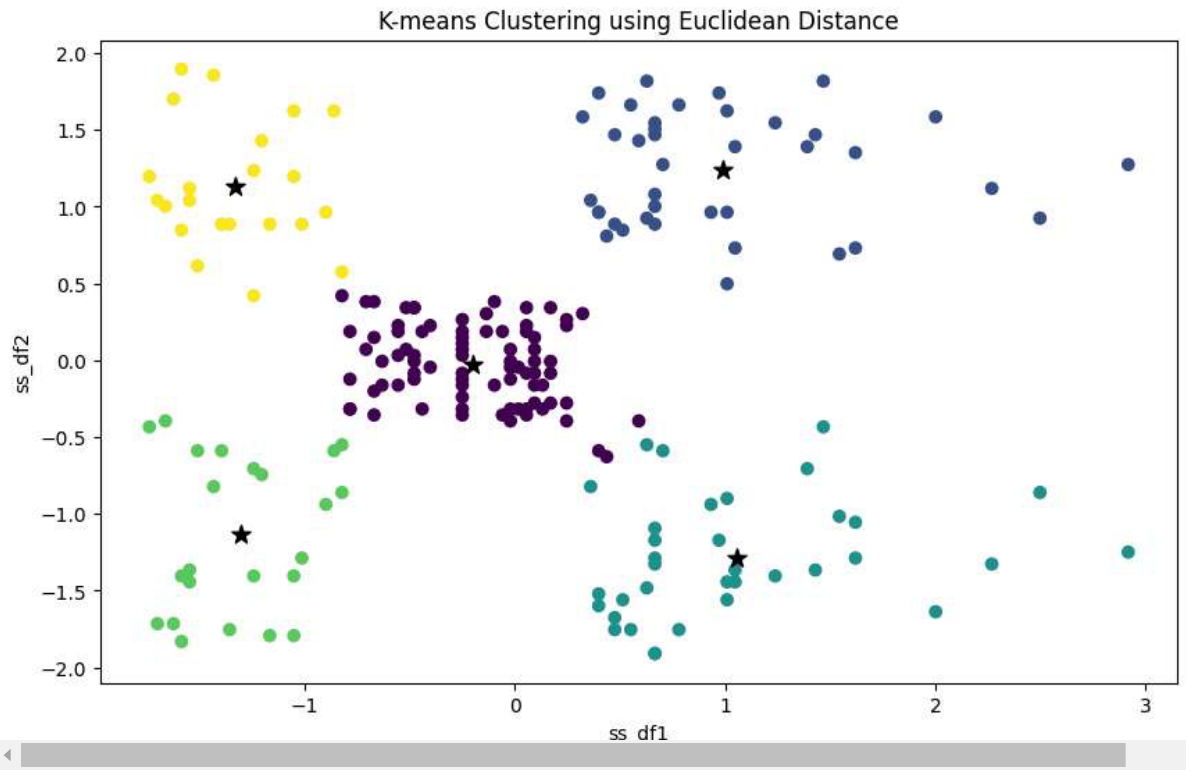
```
array([[-0.20091257, -0.02645617],
       [ 0.99158305,  1.23950275],
       [ 1.05500302, -1.28443907],
       [-1.30751869, -1.13696536],
       [-1.32954532,  1.13217788]])
```

```
l1=pd.Series(km1.labels_)
```

```
l1.value_counts()
```

	count
0	81
1	39
2	35
3	23
4	22

```
plt.figure(figsize=(10, 6))
plt.scatter(ss_df[:,0], ss_df[:,1], c=l1, cmap='viridis')
plt.scatter(cent1[:, 0],cent1[:, 1], s=100, c='black', marker='*', label='Cluster Centers')
plt.xlabel('ss_df1')
plt.ylabel('ss_df2')
plt.title('K-means Clustering using Euclidean Distance')
plt.show()
```




```
from sklearn.metrics import silhouette_score,calinski_harabasz_score

s1=silhouette_score(ss_df,l1)
c1=calinski_harabasz_score(ss_df,l1)

s1
```

```
0.5546571631111091
```

```
c1
```

```
248.64932001536357
```

```
km2=KMeans(n_clusters=3,init='k-means++',random_state=0)
km2.fit(ss_df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
```

```
KMeans(n_clusters=3, random_state=0)
```

```
cent2=km2.cluster_centers_
cent2
```

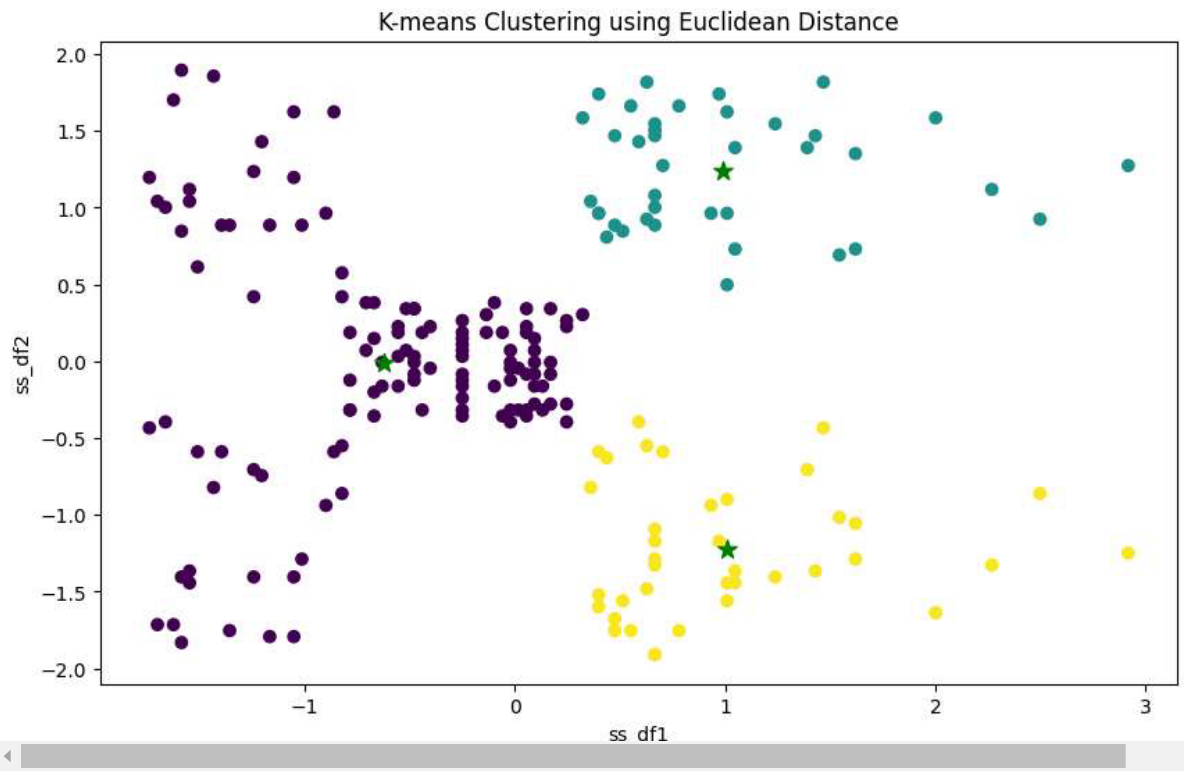
```
array([[-0.62618966, -0.01439238],
       [ 0.99158305,  1.23950275],
       [ 1.00919971, -1.22553537]])
```

```
l2=pd.Series(km2.labels_)


l2.value_counts()
```

	count
0	123
1	39
2	38

```
plt.figure(figsize=(10, 6))
plt.scatter(ss_df[:,0], ss_df[:,1], c=l2, cmap='viridis')
plt.scatter(cent2[:, 0],cent2[:, 1], s=100, c='green', marker='*', label='Cluster Centers')
plt.xlabel('ss_df1')
plt.ylabel('ss_df2')
plt.title('K-means Clustering using Euclidean Distance')
plt.show()
```




```
s2=silhouette_score(ss_df,l2)
c2=calinski_harabasz_score(ss_df,l2)
```

```
s2
```

```
0.46658474419000145
```

```
c2
```

```
151.33512126359477
```

```
# Manhattan distance : it is the measure of how apart the two data points are .
def man(X):
    return np.abs(X - np.mean(X, axis=0))
```

```
data_man = man(ss_df)
km3 = KMeans(n_clusters=5,init='k-means++',random_state=0)
km3.fit(data_man)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
```

```
KMeans(n_clusters=5, random_state=0)
```

```
cent3=km3.cluster_centers_
```

```
cent3
```

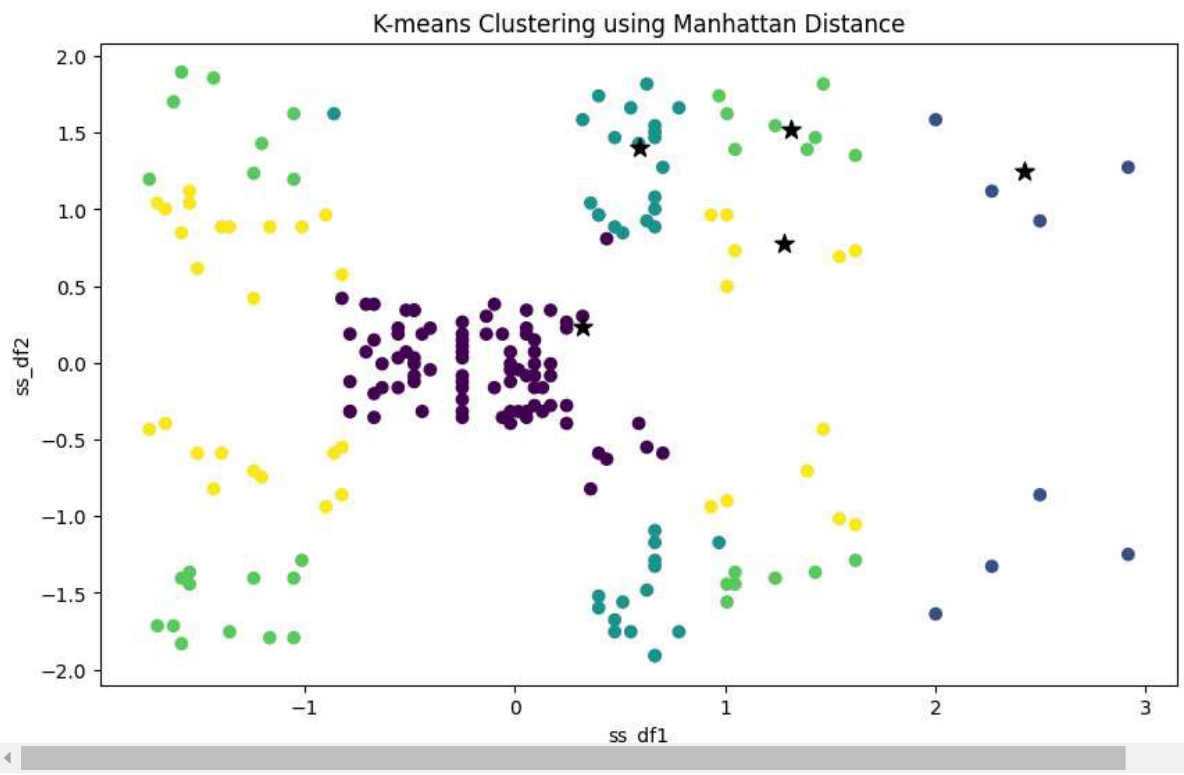
```
array([[0.3245569 , 0.2366745 ],
       [2.42146859, 1.24714264],
       [0.59052348, 1.39843889],
       [1.31390081, 1.5213615 ],
       [1.28262005, 0.77815662]])
```

```
l3=pd.Series(km3.labels_)
```

```
l3.value_counts()
```

	count
0	85
4	36
2	36
3	35
1	8

```
plt.figure(figsize=(10, 6))
plt.scatter(ss_df[:,0], ss_df[:,1], c=l3, cmap='viridis')
plt.scatter(cent3[:, 0],cent3[:, 1], s=100, c='black', marker='*', label='Cluster Centers')
plt.xlabel('ss_df1')
plt.ylabel('ss_df2')
plt.title('K-means Clustering using Manhattan Distance')
plt.show()
```




```
s3=silhouette_score(ss_df,l3)
c3=calinski_harabasz_score(ss_df,l3)
```

```
s3
```

```
0.18578482293870446
```

```
c3
```

```
10.46198967709492
```

```
#Cosine similarity is a metric used to determine how similar two vectors are
#also it measure cosine angle between them.

from sklearn.metrics.pairwise import cosine_similarity
X_cosine = cosine_similarity(ss_df)
km4 = KMeans(n_clusters=5, init='k-means++', random_state=0, algorithm='elkan')
km4.fit(X_cosine)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  super()._check_params_vs_input(X, default_n_init=10)
```

```
KMeans(algorithm='elkan', n_clusters=5, random_state=0)
```

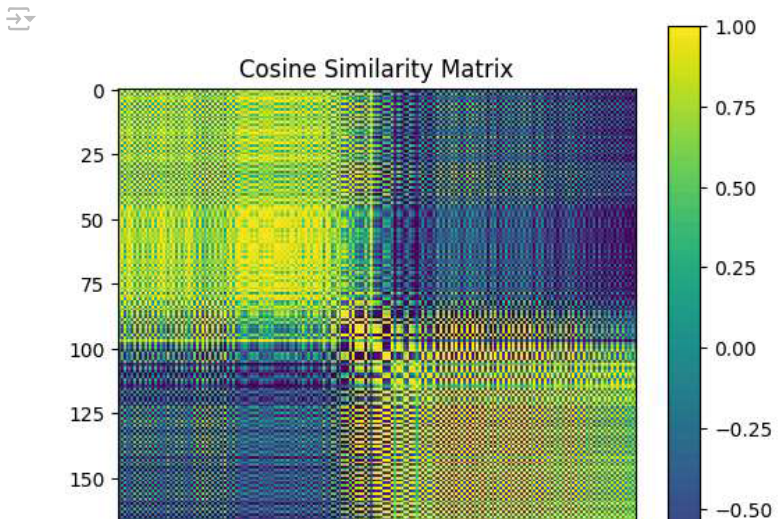
```
l4=pd.Series(km4.labels_)
```

```
l4.value_counts()
```

	count
3	49
0	48
1	45
4	35
2	23

```
# Compute cosine similarity matrix
cos_sim_matrix = np.dot(ss_df, ss_df.T) / (np.linalg.norm(ss_df, axis=1)[:, None] * np.linalg.norm(ss_df, axis=1))

# Plot cosine similarity matrix
plt.figure(figsize=(6, 6))
plt.imshow(cos_sim_matrix, cmap='viridis', interpolation='nearest')
plt.colorbar()
plt.title('Cosine Similarity Matrix')
plt.show()
```




```
s4=silhouette_score(ss_df,l4)
c4=calinski_harabasz_score(ss_df,l4)
```

```
s4
```

```
0.28087442877127583
```

+ Code + Text

```
c4
```

```
105.1267317254881
```

```
tab={'Euclidean distance with 5 cluster':{'Silhouette Score':s1,'Calinski-Harabasz Index':c1},
     'Euclidean distance with 3 cluster':{'Silhouette Score':s2,'Calinski-Harabasz Index ':c2},
     'manhattan distance ':{'Silhouette Score':s3,'Calinski-Harabasz Index':c3},
     'Cosine similarity':{'Silhouette Score':s4,'Calinski-Harabasz Index':s4}}

table=pd.DataFrame(tab)

table
```

	Euclidean distance with 5 cluster	Euclidean distance with 3 cluster	manhattan distance	Cosine similarity
Silhouette Score	0.554657	0.466585	0.185785	0.280874
Calinski-Harabasz Index	248.649320	NaN	10.461990	0.280874
Calinski-Harabasz Index	NaN	151.335121	NaN	NaN

Next steps: Generate code with table | View recommended plots | New interactive sheet

we have highest silhouette score and calinski-harabasz index for Euclidean distance with 5 cluster

```
l1_df = pd.concat([l1, df],axis=True)

l1_df
```

	0	Annual Income (k$)	Spending Score (1-100)
0	3	15	39
1	4	15	81
2	3	16	6
3	4	16	77
4	3	17	40
...	...	...	...
195	1	120	79
196	2	126	28