About our Dataset

The Mall customers dataset contains information about people visiting the mall. The dataset has gender, customer id, age, annual income, and spending score. It collects insights from the data and group customers based on their behaviors.

Our Target

Segment the customers based on the age, gender, interest. Customer segmentation is an important practise of dividing customers base into individual groups that are similar. It is useful in customised marketing.

Importing all the important libray

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
%matplotlib inline
```

Now reading our .csv file using Pandas Library

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

Out[2]:

	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

Checking for Null Values or missing Values in our Dataset

In [3]:
```
df.isna().sum()
```

Out[3]:
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Data Exploration

In [4]:
```
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 [5]:
```
df.describe()
```

Out[5]:

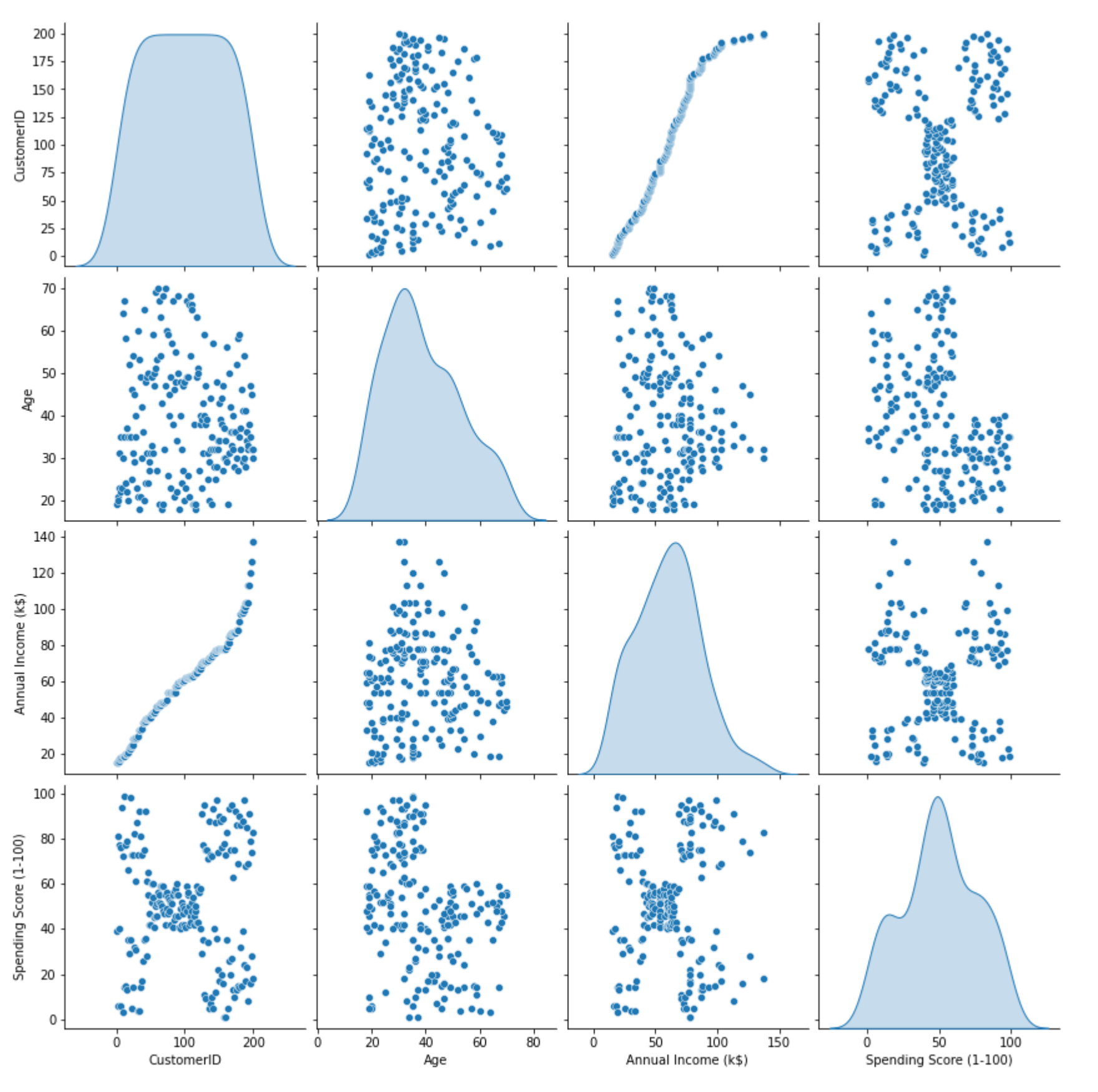
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

Plotting all the features against all other features.

We do this to get the main attributes that should be used for classifications

In [6]:
```
sns.pairplot(df  , diag_kind = 'kde' , height = 3)
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x234095fdfc0>
```



Getting the 'Correaltion' between all features

In [7]:
```
#df.iloc[:,1:].corr()
df.corr()
```

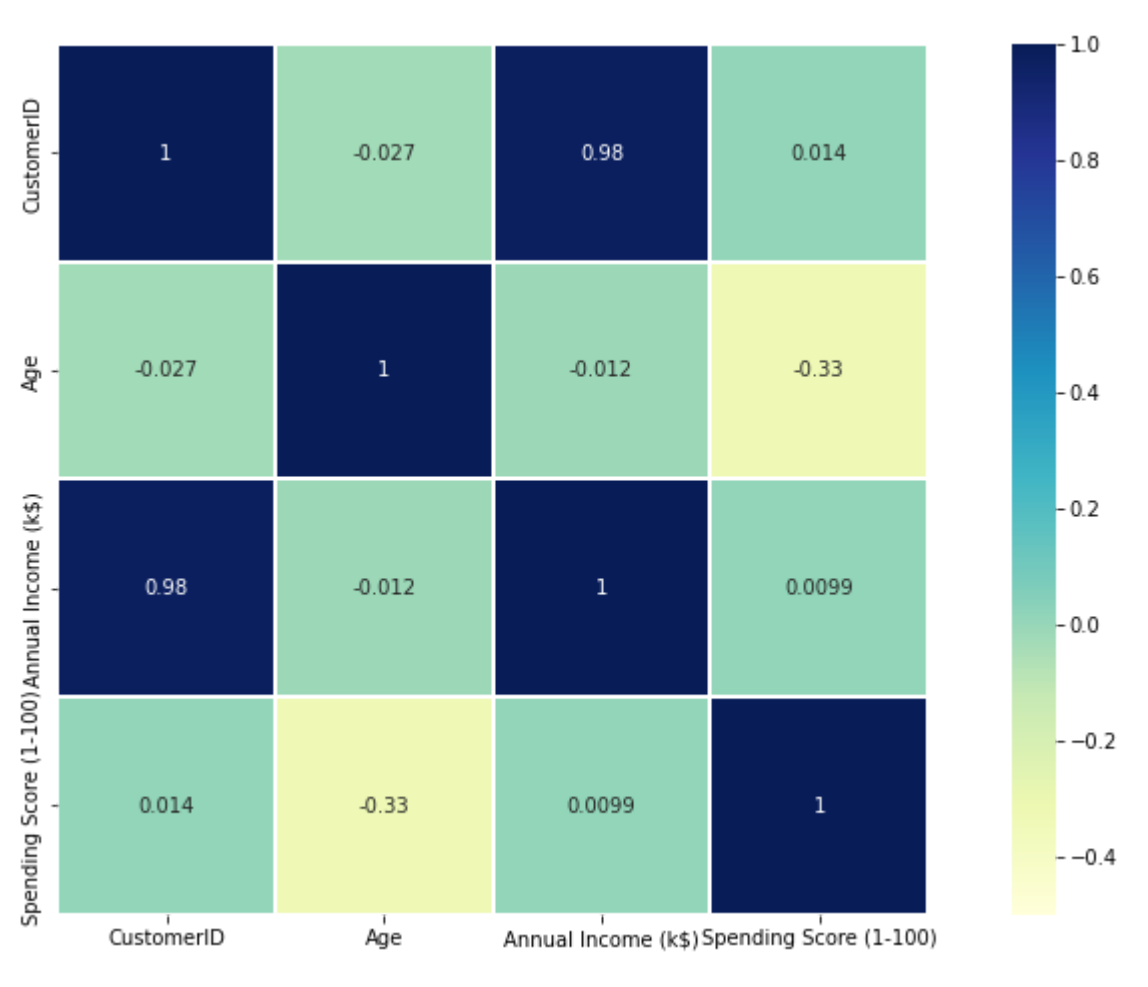
Out[7]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
CustomerID	1.000000	-0.026763	0.977548	0.013835
Age	-0.026763	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.977548	-0.012398	1.000000	0.009903
Spending Score (1-100)	0.013835	-0.327227	0.009903	1.000000

Ploting the Correlation using Heatmap

In [8]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.corr() , cmap = 'YlGnBu', annot = True , linewidths = 2, vmax = 1, vmin = -0.5 , square= True)
```

Out[8]:
```
<AxesSubplot:>
```



Removing the unneeded data in our data. Here that is Customer ID

In [9]:
```
df = df.iloc[:,1:]
df
```

Out[9]:

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

200 rows × 4 columns

Transforming the Gender or ['Genre'] coloumn into numeric data.

We are using LabelEncoding for this

In [10]:
```
from sklearn.preprocessing import LabelEncoder
encode = LabelEncoder()
df['Genre'] = encode.fit_transform(df['Genre'])
df
```

Out[10]:

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

200 rows × 4 columns

In [11]:
```
X = df.iloc[:,1:]
X
```

Out[11]:

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

200 rows × 3 columns

for Kmeans ALgo we donot need to split our dataset

In [12]:
```
'''from sklearn.model_selection import train_test_split

x_train,x_test ,y_train , y_test = train_test_split(x , y ,random_state = 5 )'''
```

Out[12]:
```
'from sklearn.model_selection import train_test_split\n\nx_train,x_test ,y_train , y_test = train_test_split(x
, y ,random_state = 5 )'
```

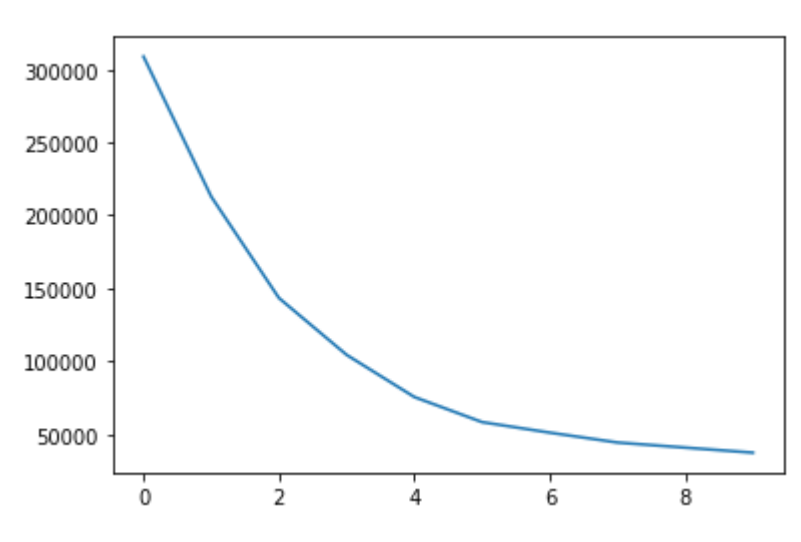
Getting the Optimum Value of K using 'Elbow Method'

In [13]:
```
from sklearn.cluster import KMeans
temp = []

for i in range(1,11):
    km = KMeans(n_clusters = i , init='k-means++')
    km.fit(X)
    temp.append(km.inertia_)
```

Ploting the Elbow Method Graph to find out the best K

In [14]:
```
plt.plot(temp)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x2340edb3b50>]
```



We get our K to be equal to '5'.

Now fitting our dataset to the Model using the best/optimal K value

In [15]:
```
kmeans = KMeans(n_clusters= 5 ,  init='k-means++')
label = kmeans.fit_predict(X)
```

Theses are the Centers of our 5 Clusters

In [16]:
```
kmeans.cluster_centers_[:]
```

Out[16]:
```
array([[32.69230769, 86.53846154, 82.12820513],
       [45.2173913 , 26.30434783, 20.91304348],
       [40.66666667, 87.75      , 17.58333333],
       [25.52173913, 26.30434783, 78.56521739],
       [43.08860759, 55.29113924, 49.56962025]])
```

The Predicted Info about which data goes to which cluster

In [17]:
```
label
```

Out[17]:
```
array([1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3,
       1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3,
       1, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 0, 2, 0, 4, 0, 2, 0, 2, 0,
       2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 4, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
       2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
       2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
       2, 0])
```

In [18]:
```
#### Adding a Coloumn next to our each data value for ease of plotting and understanding
```

In [19]:
```
X['Cluster'] = label
```

In [20]:
```
X
```

Out[20]:

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

200 rows × 4 columns

Creating a New DataFrame for the Centers of Clusters as it helps in Plotting

In [21]:
```
df2= pd.DataFrame(kmeans.cluster_centers_)
df2
```

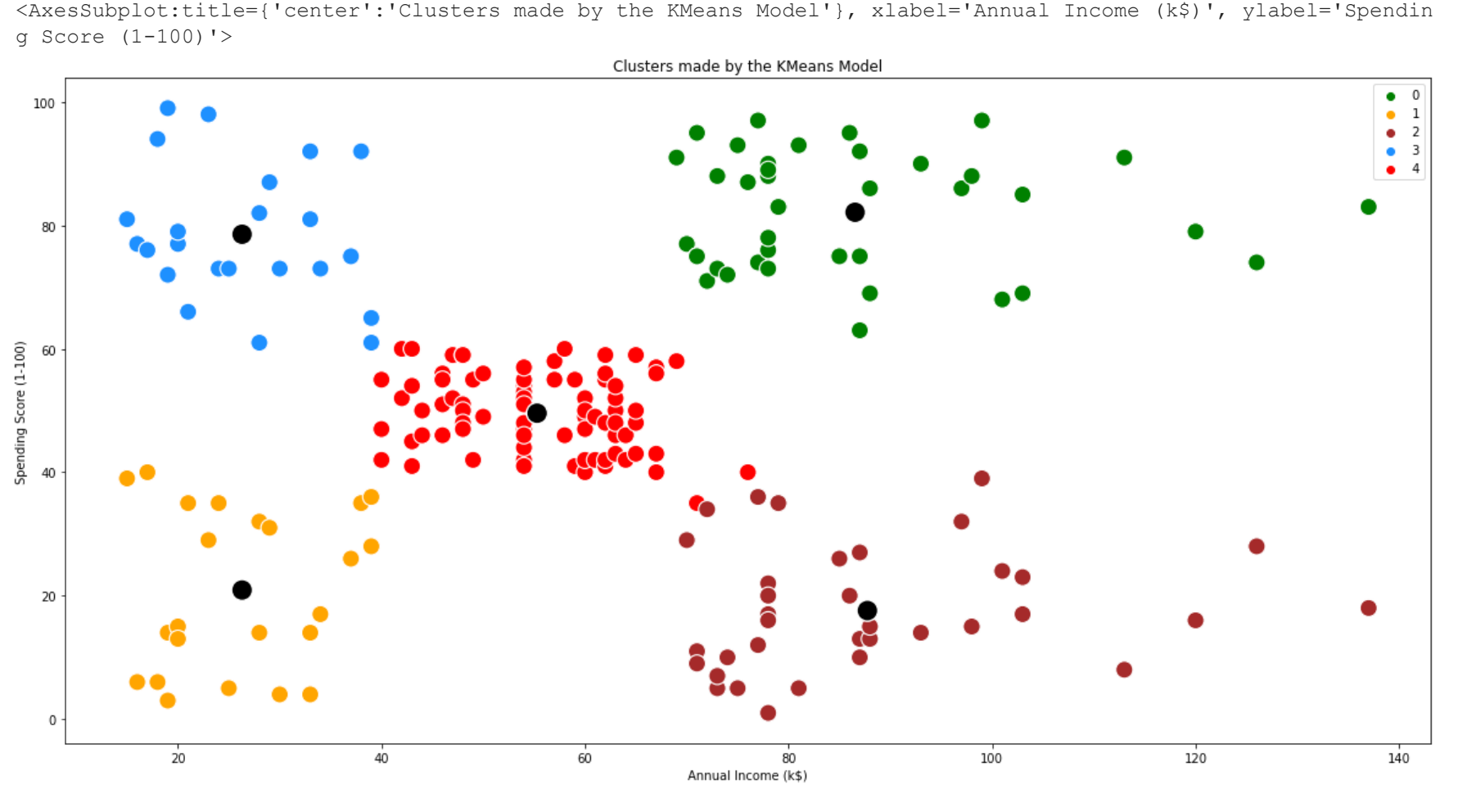
Out[21]:

	0	1	2
0	32.692308	86.538462	82.128205
1	45.217391	26.304348	20.913043
2	40.666667	87.750000	17.583333
3	25.521739	26.304348	78.565217
4	43.088608	55.291139	49.569620

PLotting Clusters in 2D

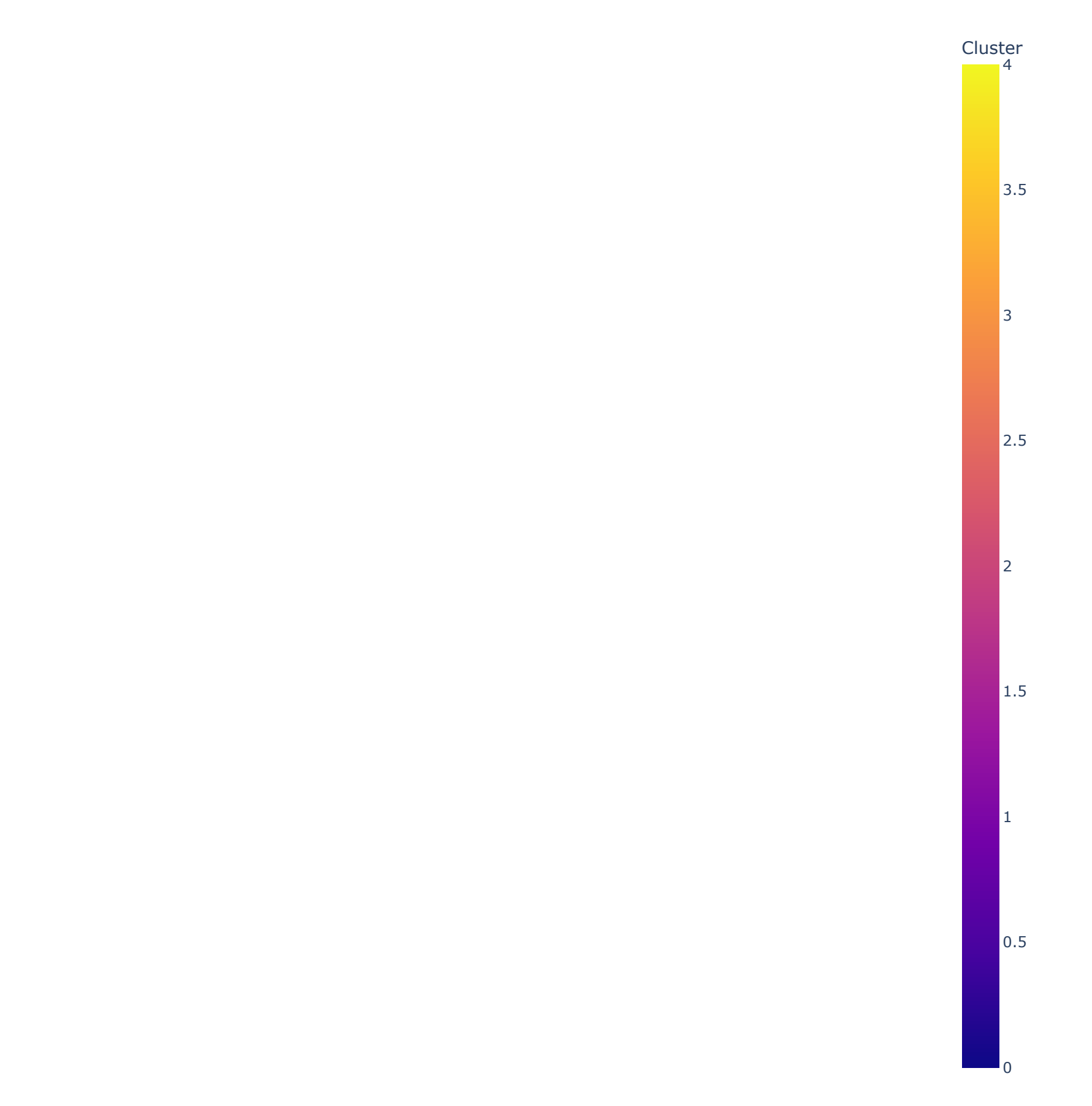
In [22]:
```
plt.figure(figsize=(20,10))
sns.scatterplot(data = X , x = 'Annual Income (k$)' ,y = 'Spending Score (1-100)',s = 200
                , hue = 'Cluster',palette=['green','orange','brown','dodgerblue','red'],
                legend='full').set(title='Clusters made by the KMeans Model')

sns.scatterplot(x = 1 , y = 2 , data = df2 , s = 300 , c = ['black','black','black','black','black'])
```

Out[22]:
```
<AxesSubplot:title={'center':'Clusters made by the KMeans Model'}, xlabel='Annual Income (k$)', ylabel='Spendin
g Score (1-100)'>
```



Ploting Clusters in 3D

In [23]:
```
px.scatter_3d( X, x = 'Annual Income (k$)' ,y = 'Spending Score (1-100)',z='Age', color = 'Cluster' , height =
```



In [ ]:

In [ ]: