Importing the necessary libraries

In [12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

let's do some data exploration to find patterns and understand the data

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

In [3]:
```
data
```

Out[3]:

	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

In [4]:
```
data.shape

#the dataset contains 200 rows and 5 columns
```

Out[4]:
```
(200, 5)
```

In [5]:
```
#data.describe, gives the summary stattistics of the datatset
data.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

In [6]:
```
#let's find out if there are null or missing values in the dataset
data.isnull().sum()
```

Out[6]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

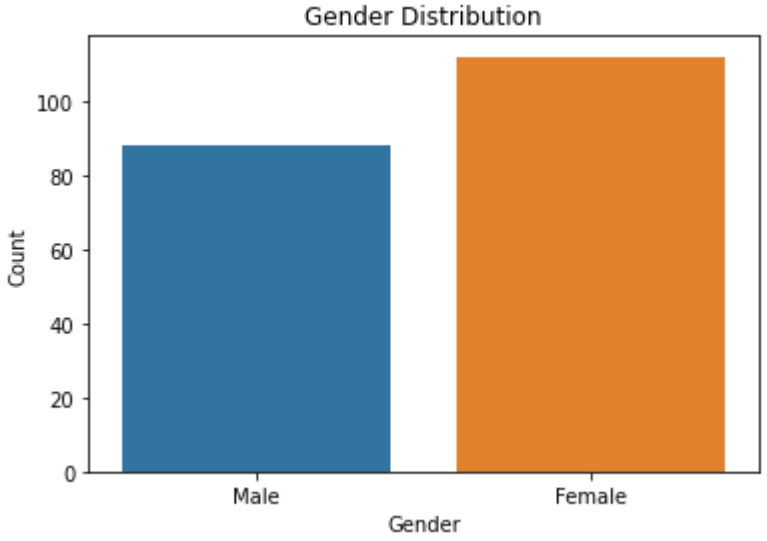
There are no null values in the dataset making it easier to work with

Now, we are about to understand better the dataset and derive and describe relationships

In [7]:
```
#visualizing the gender column to know the number of male and female custome
data["Gender"].value_counts()
```

Out[7]:
```
Female    112
Male       88
Name: Gender, dtype: int64
```

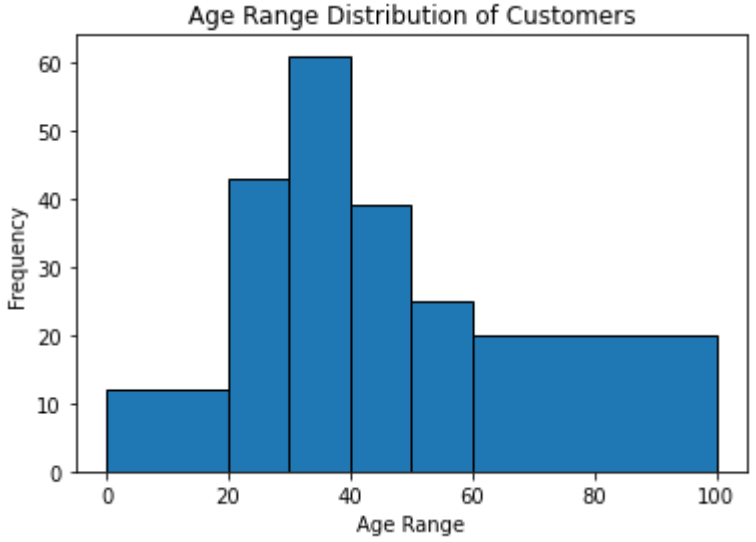
In [8]:
```
#describing graphically
sns.countplot(x='Gender', data=data)
#setting the countplot labels
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Gender Distribution')
# Show the plot
plt.show()
```



There are more female customers

In [9]:
```
bins = [0, 20, 30, 40, 50, 60, 100]  # Age bins: 0-20, 21-30, 31-40, 41-50,

# Create a histogram showing age distribution in ranges
plt.hist(data['Age'], bins=bins, edgecolor='black')
plt.xlabel('Age Range')
plt.ylabel('Frequency')
plt.title('Age Range Distribution of Customers')
plt.show()
```



According to the plot above, we can deduce that the store has more youths (aged 20 -40) as

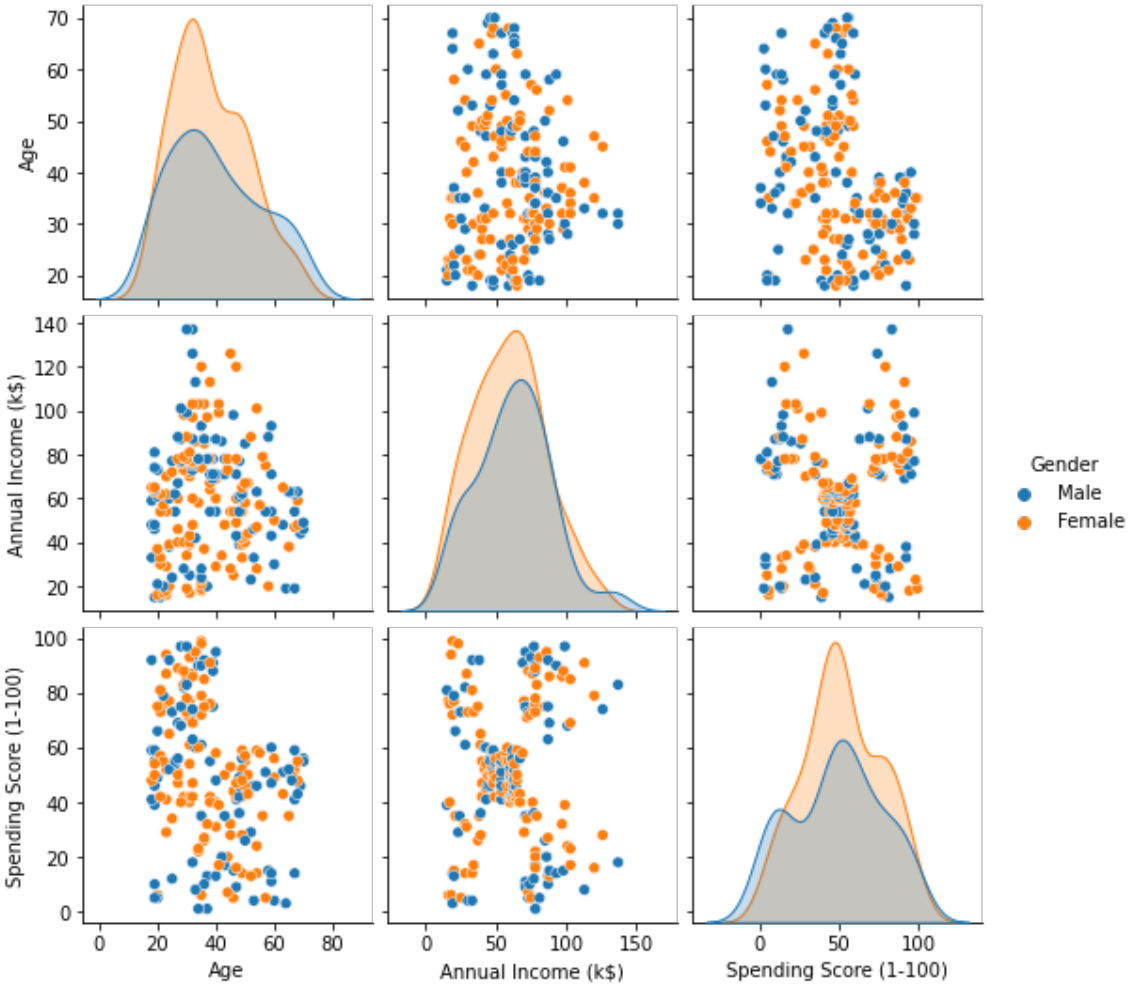
customers. Therefore, to expand their customer base among the youths, more youth-inclined products should be advertised on their website. Also, for more adult inclusion in their customer base, surveys can be made for people between age 50 -70 to know the challenges they face buying from the store for necessary actions to be taken

In [10]:
```
#comparing the annual income of customer across their various ages
#we will achieve this using lineplot

sns.lineplot(x=data["Age"], y=data["Annual Income (k$)"], data=data, marker=
plt.xlabel('Age')
plt.ylabel('Annual Income (k$)')
plt.title('Line Plot of Annual income by Age')
plt.show()
```

In [11]:
```
data = data.drop("CustomerID", axis = 1)
sns.pairplot(data, hue = "Gender")
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x2280ca48cd0>
```



In [25]:
```
data.groupby(["Gender"])["Age", "Annual Income (k$)", "Spending Score (1-10(
```

```
<ipython-input-25-20b971d3bb5b>:1: FutureWarning: Indexing with multiple k
eys (implicitly converted to a tuple of keys) will be deprecated, use a li
st instead.
  data.groupby(["Gender"])["Age", "Annual Income (k$)", "Spending Score (1
-100)"].mean()
```

Out[25]:

Gender	Age	Annual Income (k$)	Spending Score (1-100)
Female	38.098214	59.250000	51.526786
Male	39.806818	62.227273	48.511364

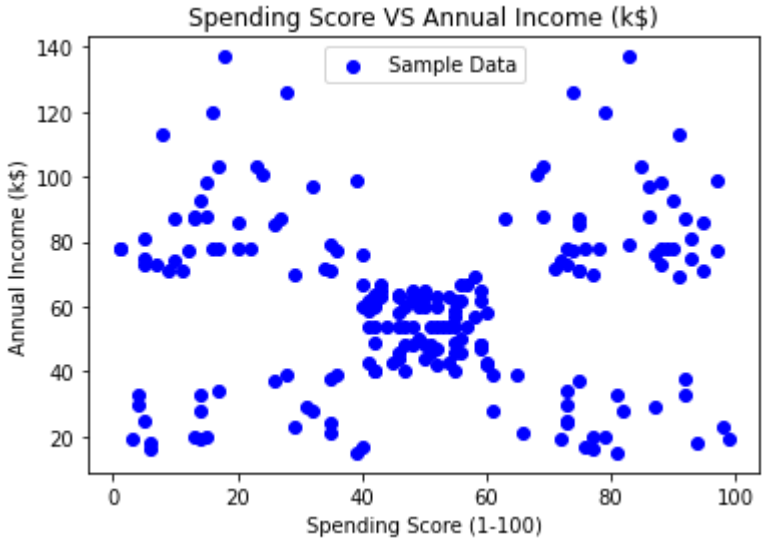
The above shows that the average income of the male customers is higher than the females' whereas the males have a lower spendng score. Due to this, we sense a negative linear relationship between the annual income of customers and their spending score,let's depict this on a scatterplot to know if we are right or not.

In [26]:
```
plt.scatter(x = data["Spending Score (1-100)"], y = data["Annual Income (k$

# Set labels and title
plt.xlabel('Spending Score (1-100)')
plt.ylabel('Annual Income (k$)')
plt.title('Spending Score VS Annual Income (k$)')

# Show legend
plt.legend()

# Show the plot
plt.show()
```



Boom! our prediction is definitely wrong, however, for clarity of the relationship, there is a need for deeper statistical analysis to identify hidden patterns or relationships in the two variables. Before we proceed, our graph depicts cluster patterns, about 5 in number. Hence, HEATMAP.....

In [27]:
```
#before plotting our heatmap, let us derive a correlation table
data.corr()
```

```
<ipython-input-27-66f400233f04>:2: FutureWarning: The default value of num
eric_only in DataFrame.corr is deprecated. In a future version, it will de
fault to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  data.corr()
```

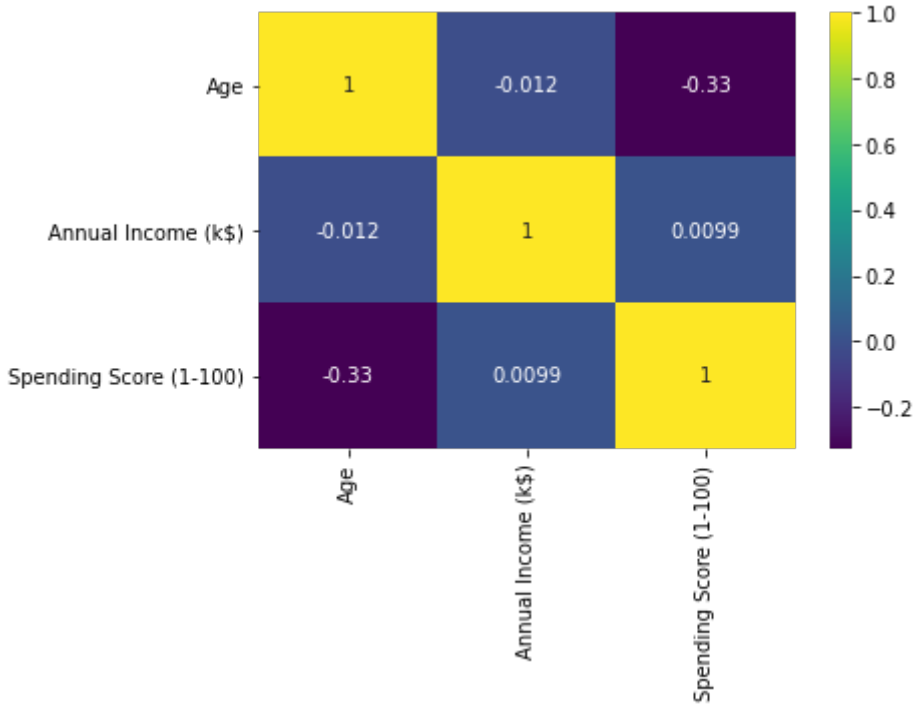
Out[27]:

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

In [28]:
```
#it is better visualized on a HEATMAP.....
sns.heatmap(data.corr(), annot = True, cmap='viridis')
```

```
<ipython-input-28-3465a89bbf3f>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(data.corr(), annot = True, cmap='viridis')
```

Out[28]: <AxesSubplot:>



Modeling

The major objective of this project is to build a KMeans model that will cluster the datasets. Based on this clusters, wise business decision can be made including recommendation and advertisement decision for the progress of the business. The model will group the varous datapoints into clusters based on similarities. However, we do not know the cluster number that will give the best silhouette score and inertia, therefore, during this course, we will determine the best value of "K" that will give the best form of clustering. In addition, new customers can also be added to clusters based on their data.

After modeling, we are expected to have achieved: 1)appropriate segmentation of the customers into groups 2)targeted marketing and product recommendation 3)improved customer experience and service 4)customer retention and market expansion

Bivariate clustering

This type of clustering involves the use of two(BI) important features that can help us achieve the best clusters or segmentation. The Annual Income and Spending Score columns are the strongest or the most important features that can aid our modeling.

In [57]:
```
###  Important imports for modeling and evaluation
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
```

Feature selection

There is a need to select the most insightful features or columns from the dataset that align with our goals. For example, the "CustomerID" column does not contain information that will be useful in the clustering process. Likewise, since we are not clustering based on gender, we will not need the "Gender" column as well.

We will be using the 'Annual Income (k$)','Spending Score (1-100)'] columns for this modeling process as they are most important features of the dataset that can tells us more about the customers and their purchasing ability.

In [58]:
```
#slicing for feature selection
X = data.iloc[:,[3,4]]
X
```

Out[58]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40
...	...	...
195	120	79
196	126	28
197	126	74
198	137	18
199	137	83

200 rows × 2 columns

Looking at these three columns, there is a wide variation in the values or datapoints, therefore, we will need to SCALE this dataset. Why do we need to scale: 1)to prevent the domination of large fetures or values which may lead to biased modeling

2)to ensure that all features contribute equally to developing the model

For example, the dataset from above has wide varaiation in values ranging from 137, 99, 18 to 1, as a result, it is pertinent to scale the values to ensure an even distribution of these features so they can contribute equally to the analysis.

In [59]:
```
X_scaled = StandardScaler().fit_transform(X)
X_scaled[:2,:]
```

Out[59]:
```
array([[-1.73899919, -0.43480148],
       [-1.73899919,  1.19570407]])
```

In [60]:
```
# Instantiate model
Xmodel = KMeans(n_clusters = 5)
Xmodel.fit(X_scaled)
data["Spending and Income Cluster"] = Xmodel.labels_
```

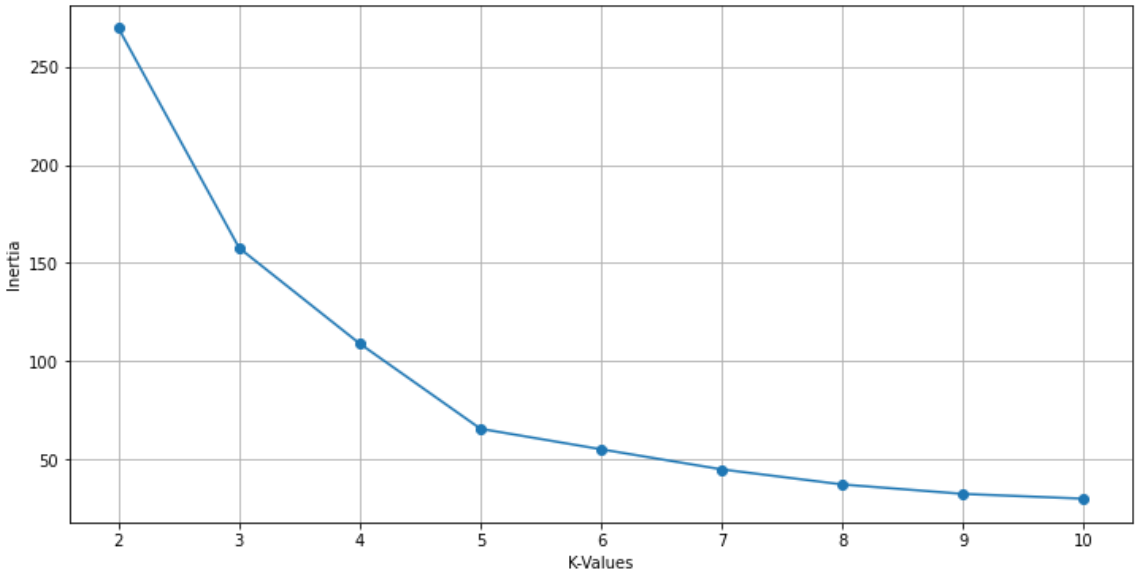
In [61]:
```

Xmodel.labels_
```

Out[61]:
```
array([1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 0,
       1, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 2, 0, 2, 3, 2, 3, 2,
       0, 2, 3, 2, 3, 2, 3, 2, 3, 2, 0, 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, 2])
```

We need to compare the inertias of multiple k values. To do this, we'll create a function that fits a K-means model for multiple values of k, calculates the inertia for each k value, and appends it to a list.

In [45]:
```
inertia_score = []
for i in range(2,11):
    kmeans = KMeans(n_clusters=i, init = "k-means++")
    kmeans.fit(X_scaled)
    inertia_score.append(kmeans.inertia_)
plt.figure(figsize = (12,6))
plt.grid()
plt.xlabel("K-Values")
plt.ylabel("Inertia")
plt.plot(range(2,11), inertia_score, marker= "o")
plt.show()
```



This plot contains an elbow at 5 clusters. Models with more than 5 clusters don't seem to reduce inertia much at all. Right now, it seems like a 5-cluster model might be optimal.

Let's now check silhouette scores. Hopefully the results will corroborate our findings from the assessment of inertia.

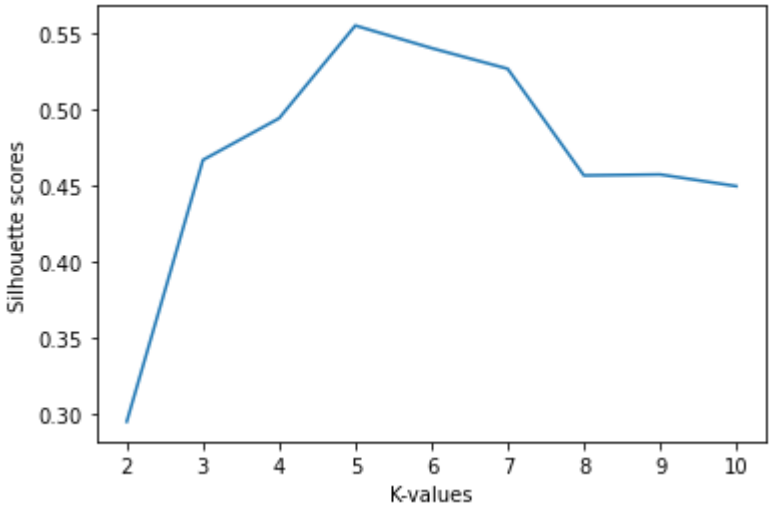
Silhouette scores

In [46]:
```
# Get silhouette score for kmeans3 model
kmeans_sil_score = silhouette_score(X_scaled, Xmodel.labels_)
kmeans_sil_score
```

Out[46]:
```
0.5546571631111091
```

It worked! However, this value isn't very useful if we have nothing to compare it to. Just as we did for inertia, we'll write a function that compares the silhouette score of each value of k, from 2 through 10.

In [47]:
```
from sklearn.metrics import silhouette_score
silhouettes = []
kmin = 2
kmax = 10
for k in range(kmin, kmax+1):
# We fit the KMeans algo with k clusters
    kmean = KMeans(n_clusters = k).fit(X_scaled)
    labels = kmean.labels_
# We calculate the silhouette score and append it to the silhouette list
    silhouettes.append(silhouette_score(X_scaled, labels, metric = 'euclide
plt.plot(range(kmin,kmax+1),silhouettes)
plt.xlabel("K-values")
plt.ylabel("Silhouette scores")
plt.show()
```



The optimal silhouette score is the highest score on the graph, meaning that at 5 clusters, we have the most optimal model. This corroborates our result on the inertia/kvalue graph. Finally, the most appropriate model has 5 clusters.

In [ ]:

Cluster visualization

In [95]:
```
x1 = data['Annual Income (k$)']
y1 =  data['Spending Score (1-100)']
X_scaled = pd.DataFrame(X_scaled)
X_scaled
```

Out[95]:

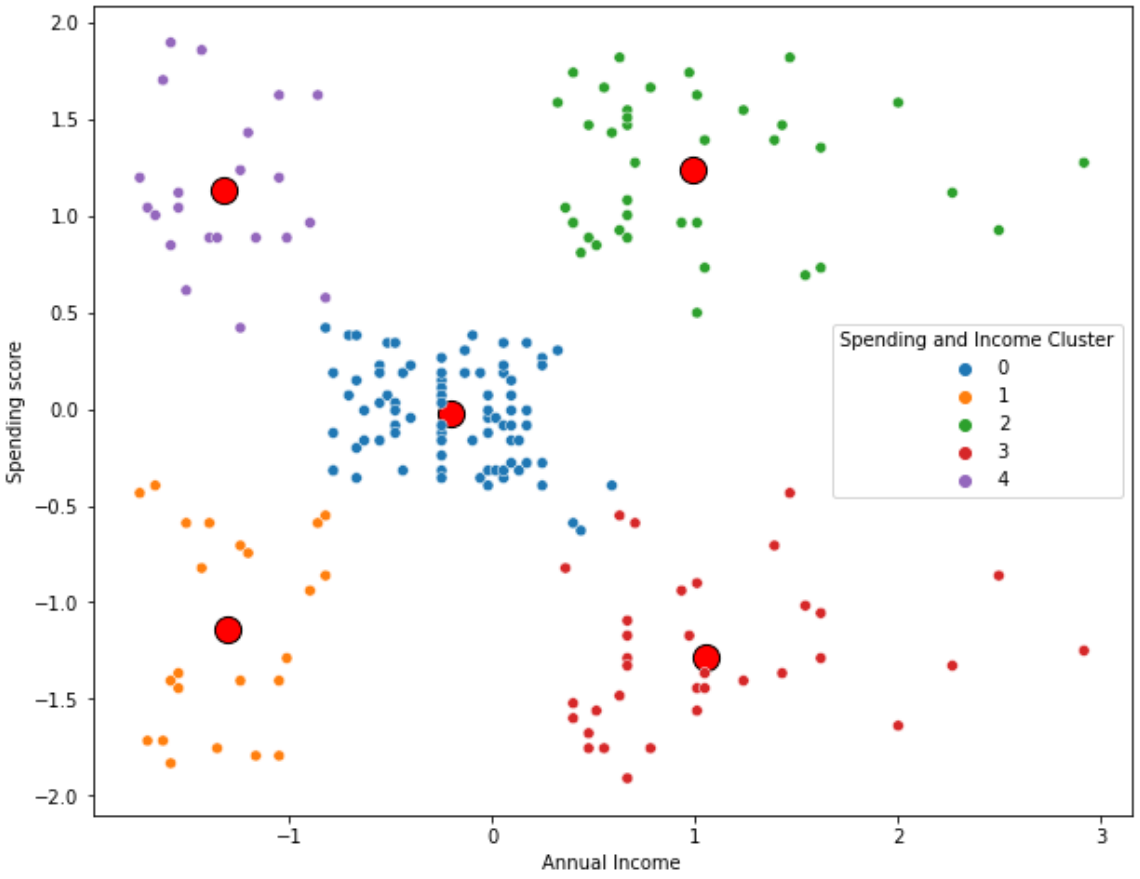
	0	1
0	-1.738999	-0.434801
1	-1.738999	1.195704
2	-1.700830	-1.715913
3	-1.700830	1.040418
4	-1.662660	-0.395980
...	...	...
195	2.268791	1.118061
196	2.497807	-0.861839
197	2.497807	0.923953
198	2.917671	-1.250054
199	2.917671	1.273347

200 rows × 2 columns

In [98]:
```
#cluster centre addition
centres = Xmodel.cluster_centers_
centres = pd.DataFrame(centres)
centres.columns = ["x", "y"]
centres

plt.figure(figsize=(10,8))
plt.scatter(centres.iloc[:, 0],centres.iloc[:, 1], c='red', marker='o', s=2(
sns.scatterplot(data = X_scaled, x = X_scaled.iloc[:,0], y =X_scaled.iloc[:
plt.xlabel("Annual Income")
plt.ylabel("Spending score")
```

Out[98]: Text(0, 0.5, 'Spending score')



The above plot shows 5 distinct clusters as predicted by the two initial graphs. Personalized advertisement or business decisions can be made accordingly. Cluster interpretation:

1)If there are going to be personalized end-of-the-year gifts and discounts, customers in cluster 2 should be taken into consideration before others, in addition, they should earn higher reward. Reason being that they have the highest spending scores. Those in this cluster are also characterized by higher annual income.

2)More luxury goods advertisement should be directed at those in cluster 2.

3)Inferior/low cost goods promotions should be advertised or sent to email list of those in clusters 1 and 3; they have low incomes and low spending score.

4)customer satisfaction survey should be specifically designed/conducted for those in cluster 3, these ones have low spending scores despite earning high.

5)Of notable mention are those in cluster 4, despite the fact that they have low annual income, they have high spending score, appropriate discount and promos should be