CUSTOMER SEGMENTATION

DATA SCIENCE INTERNSHIP

EXPOSYS DATA LABS

AVINASH PATHY

R.V. COLLEGE OF ENGINEERING, BANGALORE

TABLE OF CONTENTS

INTRODUCTION

Brief introduction to Customer Segmentation

Customer Segmentation is the subdivision of a market into discrete customer groups that share similar characteristics. Customer Segmentation can be a powerful means to identify unsatisfied customer needs. Using the above data companies can then outperform the competition by developing uniquely appealing products and services.

The most common ways in which businesses segment their customer base are:

Demographic information, such as gender, age, familial and marital status, income, education, and occupation.

Geographical information, which differs depending on the scope of the company. For localized businesses, this info might pertain to specific towns or counties. For larger companies, it might mean a customer's city, state, or even country of residence.

Psychographics, such as social class, lifestyle, and personality traits.

Behavioral data, such as spending and consumption habits, product/service usage, and desired benefits.

Need of Customer Segmentation

1. Determine appropriate product pricing.
2. Develop customized marketing campaigns.
3. Design an optimal distribution strategy.
4. Choose specific product features for deployment.
5. Prioritize new product development efforts.

How to segment customers?

Customer segmentation requires a company to gather specific information – data – about customers and analyze it to identify patterns that can be used to create segments.

Some of that can be gathered from purchasing information – job title, geography, products purchased, for example. Some of it might be gleaned from how the customer entered your system. An online marketer working from an opt-in email list might segment marketing messages according to the opt-in offer that attracted the customer, for example. Other information, however, including consumer demographics such as age and marital status, will need to be acquired in other ways.

Typical information-gathering methods include:

- Face-to-face or telephone interviews
- Surveys
- General research using published information about market categories
- Focus groups

The Challenge

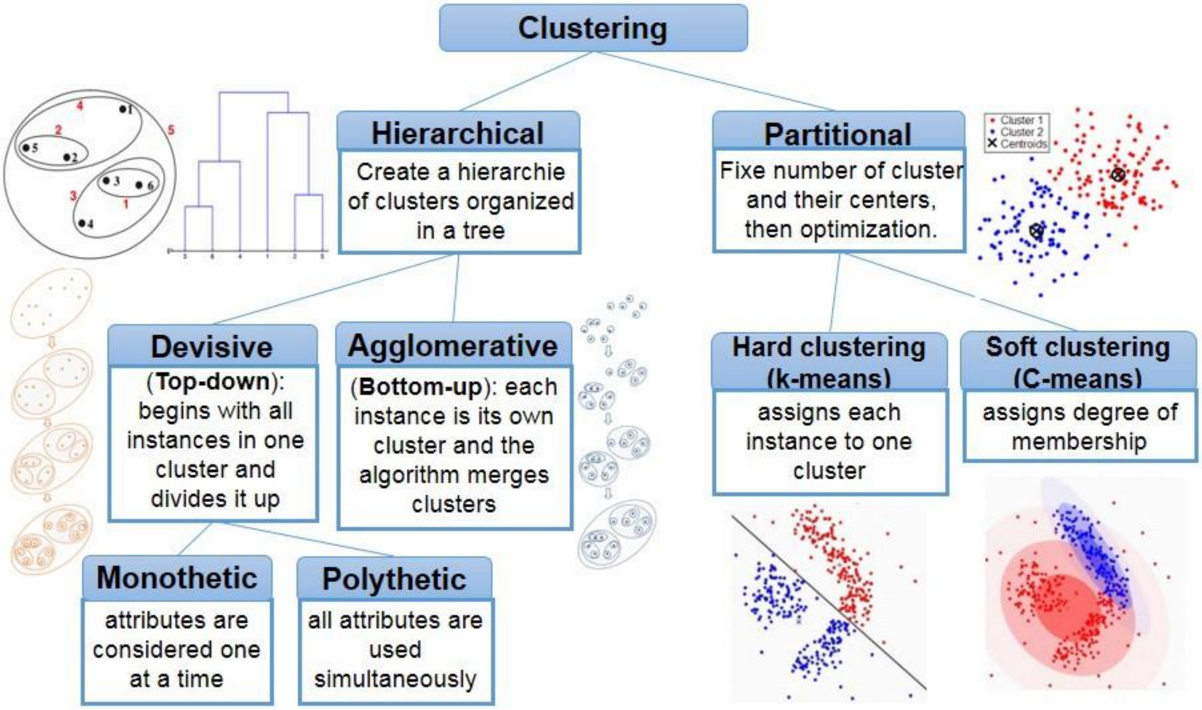
You are owing a supermarket mall and through membership cards, you have some basic data about your customers like Customer ID, age, gender, annual income and spending score. You want to understand the customers like who are the target customers so that the sense can be given to marketing team and plan the strategy accordingly.

Using Customer Segments

Common characteristics in customer segments can guide how a company markets to individual segments and what products or services it promotes to them. A small business selling hand-made guitars, for example, might decide to promote lower-priced products to younger guitarists and higher-priced premium guitars to older musicians based on segment knowledge that tells them that younger musicians have less disposable income than their older counterparts. Similarly, a meals-by-mail service might emphasize convenience to millennial customers and “tastes-like-mother-used-to-make” benefits to baby boomers.

Customer segmentation can be practiced by all businesses regardless of size or industry and whether they sell online or in person. It begins with gathering and analyzing data and ends with acting on the information gathered in a way that is appropriate and effective.

METHODOLOGY



Cluster is the collection of data objects which are similar to one another within the same group (class or category) and are different from the objects in the other clusters.

Clustering is an unsupervised learning technique in which there is predefined classes and prior information which defines how the data should be grouped or labeled into separate classes

It could also be considered as Exploratory Data Analysis (EDA) process which help us to discover hidden patterns of interest or structure in data

Clustering can also work as a standalone tool to get the insights about the data distribution or as a preprocessing step in other algorithms.

Why Clustering?

Clustering allows us to find hidden relationship between the data points in the dataset.

Examples:

1. In marketing, customers are segmented according to similarities to carry out targeted marketing.

2. Given a collection of text, we need to organize them, according to the content similarities to create a topic hierarchy

3. Detecting distinct kinds of pattern in image data (Image processing). It’s effective in biology research for identifying the underlying patterns.

K-Means Algorithm

1. Specify number of clusters *K*.
2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

Applications

1. K-means algorithm is very popular and used in a variety of applications such as market segmentation, document clustering, image segmentation and image compression, etc. The goal usually when we undergo a cluster analysis is either:
2. Get a meaningful intuition of the structure of the data we're dealing with.
3. Cluster-then-predict where different models will be built for different subgroups if we believe there is a wide variation in the behaviours of different subgroups. An example of that is clustering patients into different subgroups and build a model for each subgroup to predict the probability of the risk of having heart attack.

IMPLEMENTATION

Importing the needed libraries and displaying their versions:

In [216]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.cluster import KMeans
```

In [217]:
```
#Printing the versions of libraries
print(pd.__version__)
print(np.__version__)
print(sns.__version__)
print(sklearn.__version__)
```

```
1.0.5
1.18.1
0.10.1
0.23.1
```

Reading the customer dataset:

```
#Reading the csv file
data = pd.read_csv(r"C:\Users\Avinash\Desktop\customer-segmentation-dataset\Mall_Customers.csv")
```

Data Exploration:

Displaying first 10 rows of the dataset

In [172]:
```
#First 10 rows of the data
data.head(10)
```

Out[172]:

	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
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

Displaying last 10 rows of the dataset

```
#Last 10 rows of data
data.tail(10)
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	label
190	191	Female	34	103	23	0
191	192	Female	32	103	69	3
192	193	Male	33	113	8	0
193	194	Female	38	113	91	3
194	195	Female	47	120	16	0
195	196	Female	35	120	79	3
196	197	Female	45	126	28	0
197	198	Male	32	126	74	3
198	199	Male	32	137	18	0
199	200	Male	30	137	83	3

In [173]:
```
data.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
```

In [174]:
```
data.describe()
```

Out[174]:

	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

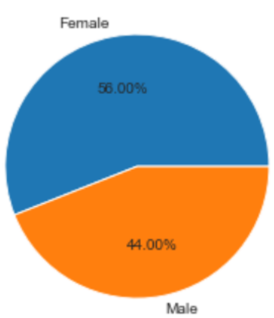
Columns present in the dataset

```
In [175]: #Columns present in the dataset
          data.columns

Out[175]: Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
                 'Spending Score (1-100)'],
                dtype='object')
```

Data Visualization:

Pie chart displaying the gender distribution

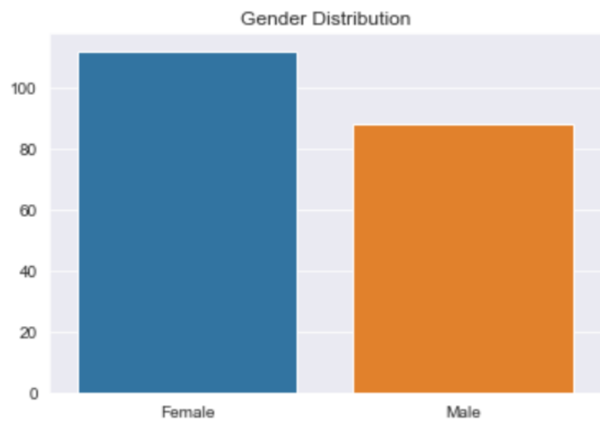
```
In [176]: #Pie chart for gender distribution
          gender = data["Gender"].value_counts()
          print(gender)
          gen = []
          gen_count = []
          for key,value in gender.items():
              gen.append(key)
              gen_count.append(value)
          #print(gen_count)
          plt.pie(gen_count, labels = gen,autopct='%1.2f%%')
          plt.show()

          Female    112
          Male       88
          Name: Gender, dtype: int64
```



Bar plot showing the number of females and males

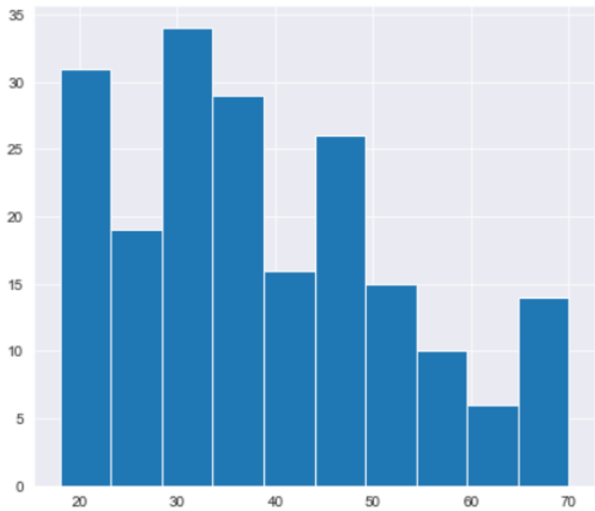
In [177]:
```
#Bar plot of gender
gender = data["Gender"].value_counts()
print(gender)
sns.set_style("darkgrid")
plt.title("Gender Distribution")
sns.barplot(x = gender.index, y = gender.values)
plt.show()
```

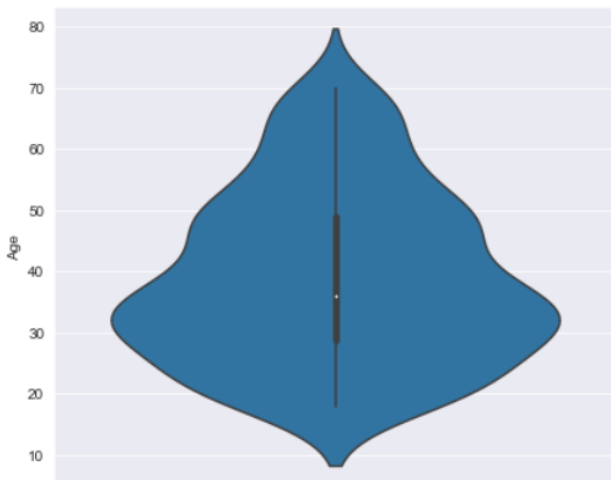
```
Female    112
Male       88
Name: Gender, dtype: int64
```



Histogram and Violin plot of age frequency

In [178]:
```
# Histogram of Age frequency
plt.figure(figsize = (15,6))
plt.xlabel("Age")
plt.ylabel("Number of People")
plt.title("Age frequency")
plt.subplot(1,2,1)
ages = data["Age"].hist()
plt.show()

#Violin Plot of Age Frequency
plt.figure(figsize = (15,6))
plt.title("Age Frequency")
sns.axes_style("dark")
plt.subplot(1,2,2)
sns.violinplot(y = data["Age"])
plt.show()
```

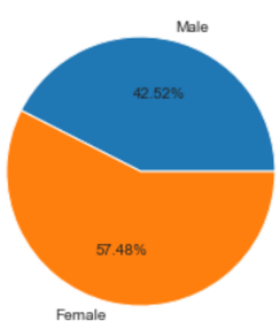




Pie chart showing total money spent by male and female

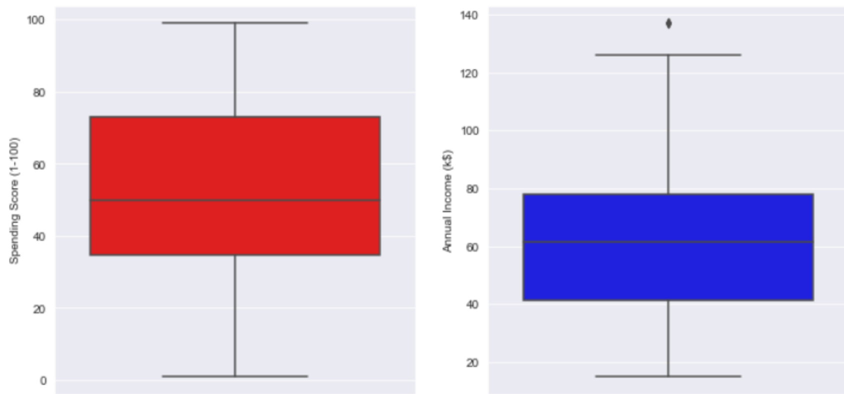
In [179]:
```
#Pie chart showing total money spent by male and female
total_spending = []
male_spending = data["Spending Score (1-100)"].where(data["Gender"]=="Male").sum()
print("Money spent by male: ",male_spending)
total_spending.append(male_spending)
female_spending = data["Spending Score (1-100)"].where(data["Gender"]=="Female").sum()
print("Money spent by female: ",female_spending)
total_spending.append(female_spending)
gender_list = ["Male","Female"]
plt.pie(total_spending, labels = gender_list,autopct='%1.2f%%')
plt.show()
```

```
Money spent by male:  4269.0
Money spent by female:  5771.0
```



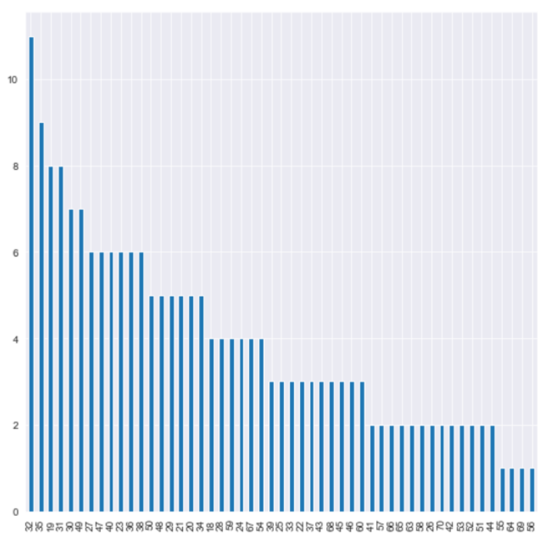
Box plot of spending score and annual income

```
In [180]: #Box plot of Spending score and Annual Income
          plt.figure(figsize = (12,6))
          plt.subplot(1,2,1)
          sns.boxplot(y = data["Spending Score (1-100)"],color = "red")
          plt.subplot(1,2,2)
          sns.boxplot(y = data["Annual Income (k$)"],color = "blue")
          plt.show()
```



Number of people of a particular age

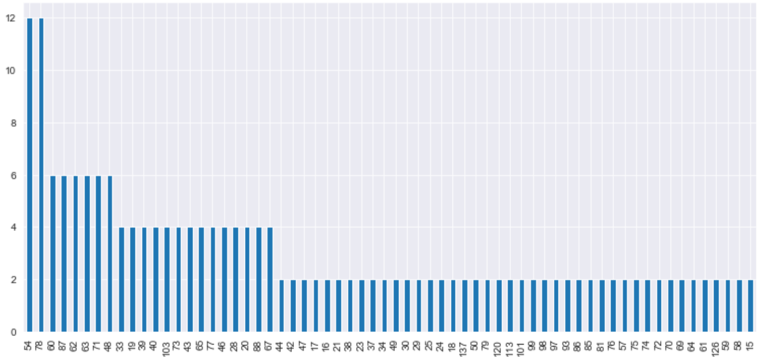
```
In [181]: data['Age'].value_counts().plot.bar(figsize = (9, 9))
Out[181]: <matplotlib.axes._subplots.AxesSubplot at 0x1de0c08fb08>
```



Number of people of having a particular annual income and spending score

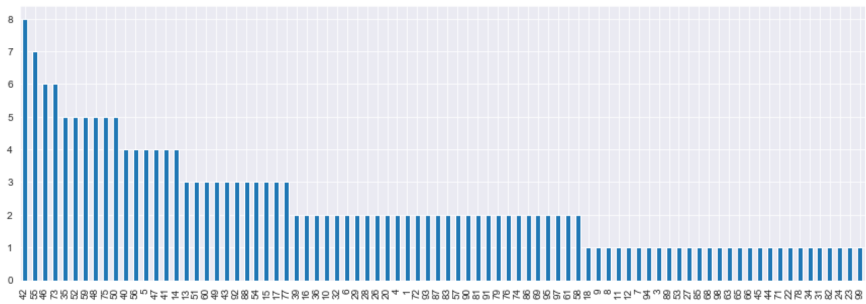
In [182]:
```python
data['Annual Income (k$)'].value_counts().plot.bar(figsize = (13, 6))
```

Out[182]: <matplotlib.axes._subplots.AxesSubplot at 0x1de0be5ee88>



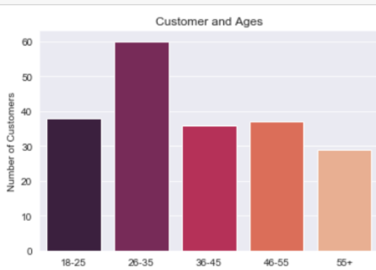
In [183]:
```python
data['Spending Score (1-100)'].value_counts().plot.bar(figsize = (15, 5))
```

Out[183]: <matplotlib.axes._subplots.AxesSubplot at 0x1de0d31b9c8>



Different age groups with bar plot

In [184]:
```python
#Different Age groups
age18_25 = data.Age[(data.Age<=25) & (data.Age>=18)]
age26_35 = data.Age[(data.Age<=35) & (data.Age>=26)]
age36_45 = data.Age[(data.Age<=45) & (data.Age>=36)]
age46_55 = data.Age[(data.Age<=55) & (data.Age>=46)]
age_above55 = data.Age[data.Age>=56]

age_group = ["18-25","26-35","36-45","46-55","55+"]
number_of_people = [len(age18_25.values),len(age26_35.values),len(age36_45.values),len(age46_55.values),len(age_above55.values)]
sns.barplot(x = age_group, y = number_of_people,palette = "rocket")
plt.title("Customer and Ages")
plt.ylabel("Number of Customers")
plt.xlabel("Ages")
plt.show()
```

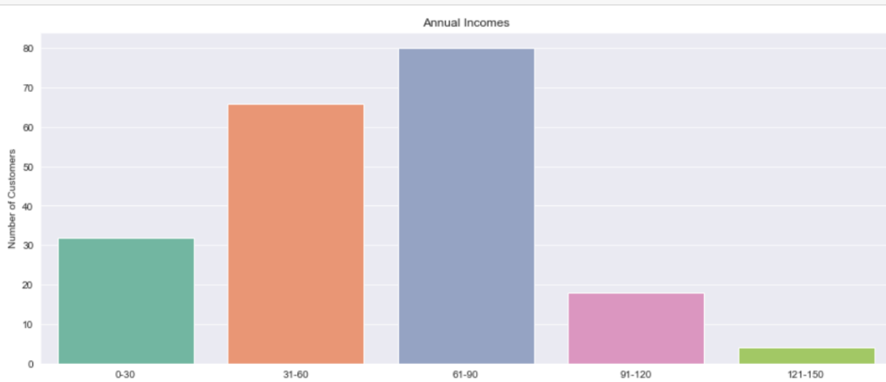


Different spending scores with bar plot

In [185]:
```python
#Grouping spending scores
sscore1_20 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"]>=1) & (data["Spending Score (1-100)"]<=20) ]
sscore21_40 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"]>=21) & (data["Spending Score (1-100)"]<=40) ]
sscore41_60 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"]>=41) & (data["Spending Score (1-100)"]<=60) ]
sscore61_80 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"]>=61) & (data["Spending Score (1-100)"]<=80) ]
sscore81_100 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"]>=81) & (data["Spending Score (1-100)"]<=100) ]
s_scorex = ["1-20","21-40","41-60","61-80","81-100"]
s_scorey = [len(sscore1_20.values),len(sscore21_40.values),len(sscore41_60.values),len(sscore61_80.values),len(sscore81_100.values)]

#Bar plot of spending score
plt.figure(figsize = (15,6))
sns.barplot(x = s_scorex, y = s_scorey,palette = "deep")
plt.title("Spending Scores")
plt.ylabel("Number of Customers having the score")
plt.xlabel("Scores")
plt.show()
```

Different annual incomes with bar plot

In [186]:
```python
#Grouping Annual Incomes
ai0_30 = data["Annual Income (k$)"][(data["Annual Income (k$)"]>=0) & (data["Annual Income (k$)"]<=30)]
ai31_60 = data["Annual Income (k$)"][(data["Annual Income (k$)"]>=31) & (data["Annual Income (k$)"]<=60)]
ai61_90 = data["Annual Income (k$)"][(data["Annual Income (k$)"]>=61) & (data["Annual Income (k$)"]<=90)]
ai91_120 = data["Annual Income (k$)"][(data["Annual Income (k$)"]>=91) & (data["Annual Income (k$)"]<=120)]
ai121_150 = data["Annual Income (k$)"][(data["Annual Income (k$)"]>=121) & (data["Annual Income (k$)"]<=150)]

aix = ["0-30","31-60","61-90","91-120","121-150"]
aiy = [len(ai0_30.values),len(ai31_60.values),len(ai61_90.values),len(ai91_120.values),len(ai121_150.values)]

#Bar plot of Annual Incomes
plt.figure(figsize = (15,6))
sns.barplot(x = aix, y = aiy,palette = "Set2")
plt.title("Annual Incomes")
plt.ylabel("Number of Customers")
plt.xlabel("Income")
plt.show()
```



K-Means Clustering Algorithm

Determining optimal number of clusters:

The "Elbow" Method:

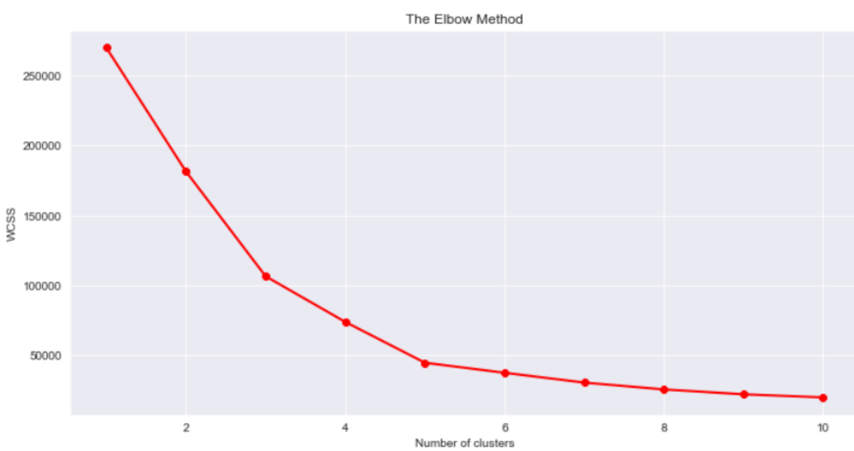
The main goal behind cluster partitioning methods like k-means is to define the clusters such that the intra-cluster variation stays minimum.

minimize(sum W(Ck)), k=1…k

Where Ck represents the kth cluster and W(Ck) denotes the intra-cluster variation.

Probably the most well-known method, the elbow method, in which the sum of squares at each number of clusters is calculated and graphed, and the user looks for a change of slope from steep to shallow (an elbow) to determine the optimal number of clusters. This method is inexact, but still potentially helpful.

In [187]:
```
#Finding optimal number of clusters using elbow method
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(data.iloc[:,[3,4]].values)
    wcss.append(kmeans.inertia_)
plt.figure(figsize = (12,6))
plt.plot(range(1, 11), wcss,color = "red",marker = "8",linewidth = "2")
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



The Elbow Curve method is helpful because it shows how increasing the number of the clusters contribute separating the clusters in a meaningful way, not in a marginal way. The bend indicates that additional clusters beyond the fifth have little value.

The Elbow method is fairly clear, if not a naïve solution based on intra-cluster variance.

The Silhouette Method

Another visualization that can help determine the optimal number of clusters is called the silhouette method. Average silhouette method computes the average silhouette of observations for different values of k. The optimal number of clusters k is the one that maximize the average silhouette over a range of possible values for k.

```
In [221]: #Finding optimal number of clusters using Average Silhouette Method
          from sklearn.metrics import silhouette_score
          range_n_clusters = list (range(2,10))
          for n_clusters in range_n_clusters:
              clusterer = KMeans(n_clusters=n_clusters)
              preds = clusterer.fit_predict(data.iloc[:,[3,4]].values)
              centers = clusterer.cluster_centers_

              score = silhouette_score(data.iloc[:,[3,4]].values, preds)
              print("For n_clusters = {}, silhouette score is {})".format(n_clusters, score))

          For n_clusters = 2, silhouette score is 0.2968969162503008)
          For n_clusters = 3, silhouette score is 0.46761358158775435)
          For n_clusters = 4, silhouette score is 0.4931963109249047)
          For n_clusters = 5, silhouette score is 0.553931997444648)
          For n_clusters = 6, silhouette score is 0.53976103063432)
          For n_clusters = 7, silhouette score is 0.5270287298101395)
          For n_clusters = 8, silhouette score is 0.45492755850983463)
          For n_clusters = 9, silhouette score is 0.4607224274992025)
```

This suggests an optimal of 5 clusters.

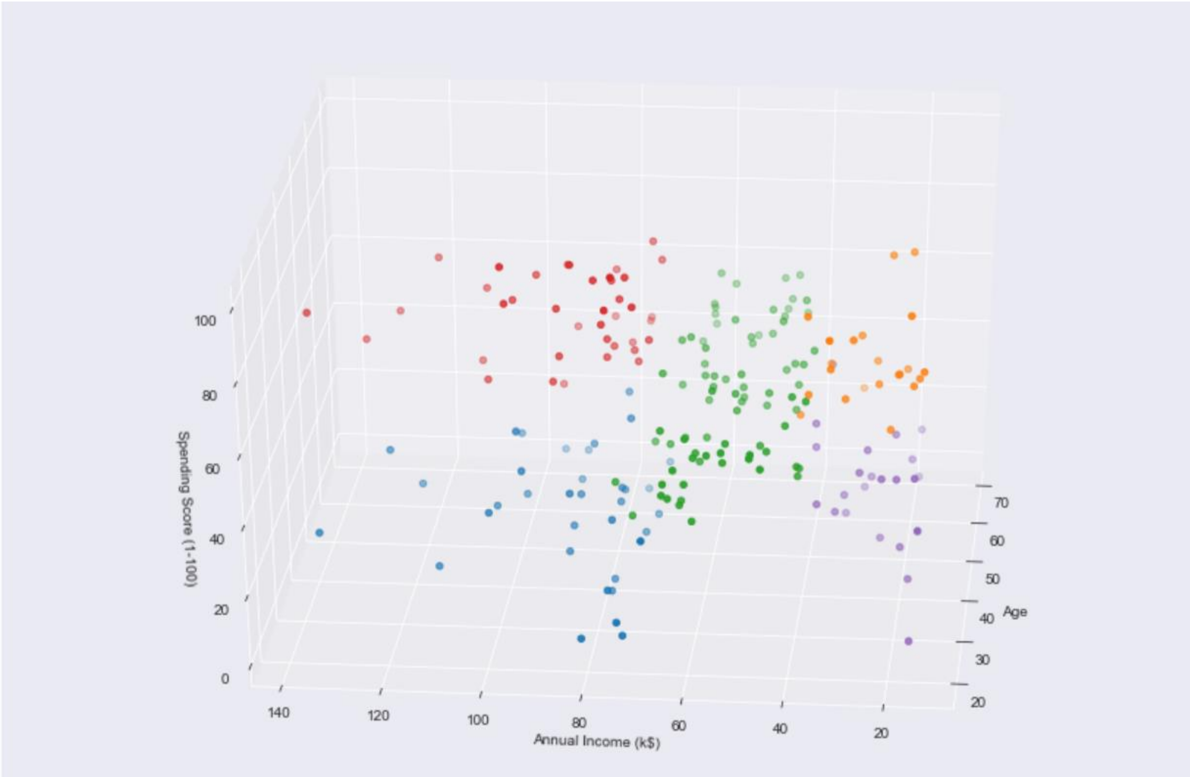
Visualizing the identified cluster:

Based on the above technique outcomes along with trial and error, it is decided to take 5 clusters for this data set

```
#Fitting k-means to the dataset
kmean = KMeans(n_clusters = 5)
clusters = kmean.fit_predict(data.iloc[:,[3,4]].values)
data["label"] = clusters
```

```
# Visualising the clusters
plt.scatter(data.iloc[:,[3,4]].values[clusters == 0, 0], data.iloc[:,[3,4]].values[clusters == 0, 1], s = 70, c = 'red')
plt.scatter(data.iloc[:,[3,4]].values[clusters == 1, 0], data.iloc[:,[3,4]].values[clusters == 1, 1], s = 70, c = 'green')
plt.scatter(data.iloc[:,[3,4]].values[clusters == 2, 0], data.iloc[:,[3,4]].values[clusters == 2, 1], s = 70, c = 'blue')
plt.scatter(data.iloc[:,[3,4]].values[clusters == 3, 0], data.iloc[:,[3,4]].values[clusters == 3, 1], s = 70, c = 'magenta')
plt.scatter(data.iloc[:,[3,4]].values[clusters == 4, 0], data.iloc[:,[3,4]].values[clusters == 4, 1], s = 70, c = 'purple')
plt.scatter(kmean.cluster_centers_[:, 0], kmean.cluster_centers_[:, 1], s = 150, c = 'orange', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```


```
# 3D Visualization
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize = (15,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data.Age[data.label==0], data["Annual Income (k$)"][data.label==0],data["Spending Score (1-100)"][data.label==0])
ax.scatter(data.Age[data.label==1], data["Annual Income (k$)"][data.label==1],data["Spending Score (1-100)"][data.label==1])
ax.scatter(data.Age[data.label==2], data["Annual Income (k$)"][data.label==2],data["Spending Score (1-100)"][data.label==2])
ax.scatter(data.Age[data.label==3], data["Annual Income (k$)"][data.label==3],data["Spending Score (1-100)"][data.label==3])
ax.scatter(data.Age[data.label==4], data["Annual Income (k$)"][data.label==4],data["Spending Score (1-100)"][data.label==4])
ax.view_init(30,185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel("Spending Score (1-100)")
plt.show()
```



CONCLUSION

The customers were divided into 5 groups as listed below.

Cluster 1(Blue) – This cluster consists of customers with a medium annual income and a medium spending score.

Cluster 2(Purple) – This cluster consists of customers with a low annual income and a low spending score.

Cluster 3 (Green)– This cluster consists of customers with a low annual income and a high spending score.

Cluster 4 (Magenta)– This cluster consists of customers with a high annual income and a high spending score.

Cluster 5(Red) – This cluster consists of customers with a high income and a low spending score.

The second and third groups consisting of low-income customers form the most minor part of the stores customers set and so they need not be given the most importance.
The store must concentrate their business strategies keeping it around the first and fourth group to gain maximum profits.