

# CUSTOMER SEGMENTATION USING DATA SCIENCE

## INTRODUCTION:

Customer segmentation using data science is a powerful technique that allows businesses to divide their customer base into distinct groups based on common characteristics, behaviors, or preferences. It involves analyzing large amounts of data, such as customer demographics, purchase history, online behavior, and social media interactions, to identify patterns and correlations.

The goal of customer segmentation is to gain a deeper understanding of different customer segments, their needs, and their value to the business. By segmenting customers, businesses can tailor their marketing strategies, products, and services to specific segments, leading to more effective targeting, improved customer satisfaction, and increased profitability.

Data science techniques, such as clustering algorithms, machine learning algorithms, and predictive modeling, are used to identify meaningful customer segments. These techniques analyze the data, detect patterns, and create models that assign customers to various segments based on similarities.

Once customer segments are identified, businesses can personalize their marketing campaigns, develop targeted promotions, provide customized product recommendations, and enhance the overall customer experience. By understanding the unique characteristics of each segment, businesses can deliver more relevant messaging, improve customer retention, and ultimately drive revenue growth.

In summary, customer segmentation using data science is an essential tool for businesses to effectively understand and cater to the diverse needs of their customer base. It helps optimize marketing efforts, improve customer satisfaction, and drive business success.

## GIVEN DATASET:

	A	B	C	D	E	F
1	CustomerID	Genre	Age	Annual Income (	Spending Score (1-100)	
2	1	Male	19	15	39	
3	2	Male	21	15	81	
4	3	Female	20	16	6	
5	4	Female	23	16	77	
6	5	Female	31	17	40	
7	6	Female	22	17	76	
8	7	Female	35	18	6	
9	8	Female	23	18	94	
10	9	Male	64	19	3	
11	10	Female	30	19	72	
12	11	Male	67	19	14	
13	12	Female	35	19	99	
14	13	Female	58	20	15	
15	14	Female	24	20	77	
16	15	Male	37	20	13	
17	16	Male	22	20	79	
18	17	Female	35	21	35	
19	18	Male	20	21	66	

## PROBLEM DEFINITION AND DESIGN THINKING:

Certainly, here's a structured outline for a customer segmentation project using data science, encompassing the problem definition, design thinking, problem goals, project objectives, and algorithmic steps:

### Problem Definition:

- **Business Context:** Start by understanding the business context and why customer segmentation is necessary. For example, a retail business might want to enhance its marketing strategies.
- **Challenges:** Identify the current challenges, such as generic marketing efforts, high customer acquisition costs, or low customer retention.

### Design Thinking:

- **Empathize:** Put yourself in the shoes of the customers to understand their needs, preferences, and pain points.
- **Define:** Clearly articulate the problem based on customer insights. In our retail example, it might be that customers receive irrelevant product recommendations.
- **Ideate:** Brainstorm potential solutions and data-driven approaches, such as segmentation, to address the problem.
- **Prototype:** Create a plan for implementing customer segmentation to solve the problem.

### Problem Goals:

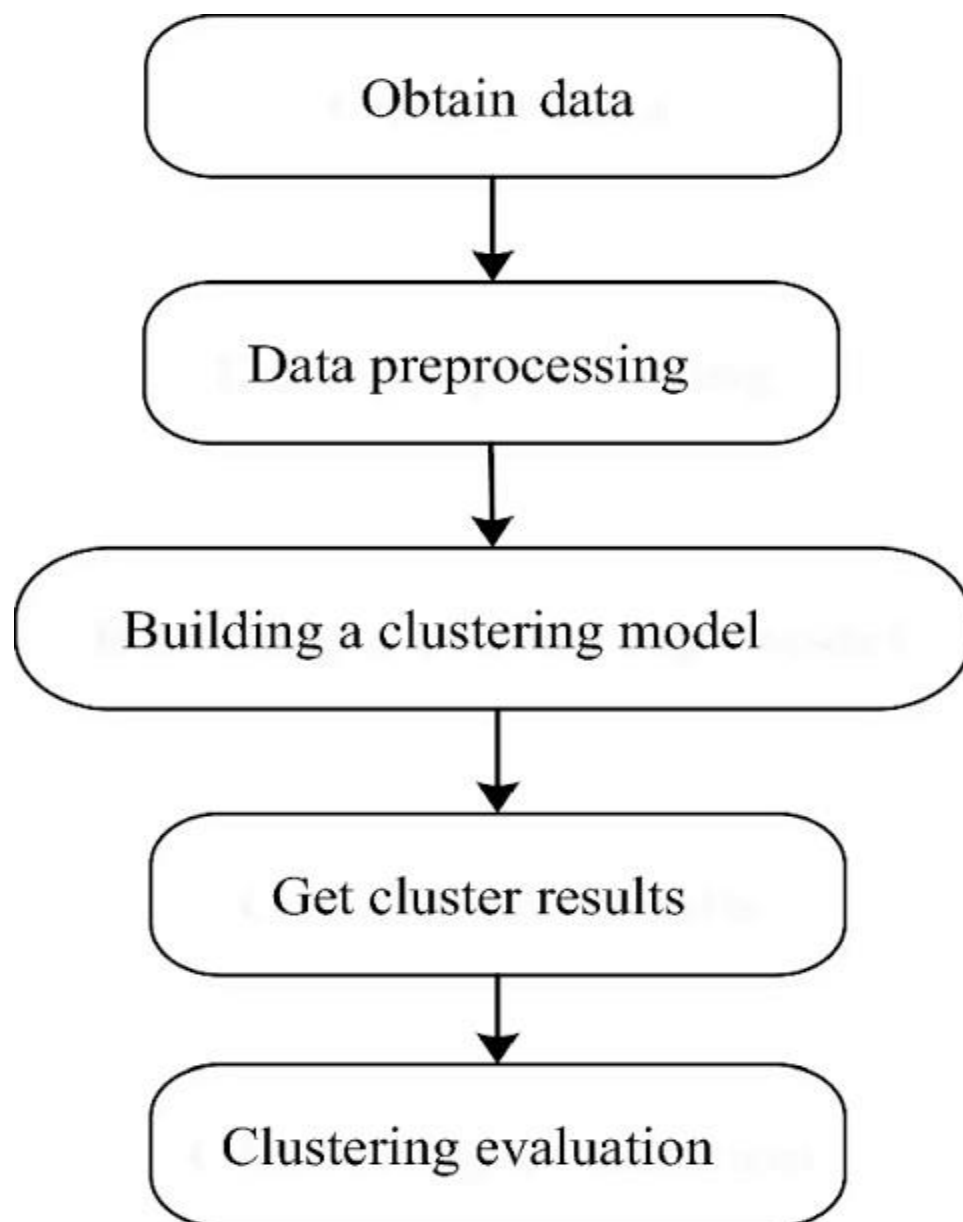
- **Customer Segmentation:** The primary goal is to segment the customer base into distinct groups based on their characteristics and behaviors.
- **Personalization:** Develop strategies for more personalized marketing, product recommendations, and customer experiences.

- **Cost Reduction:** Optimize marketing spending by targeting the right segments with the right messages, reducing costs.

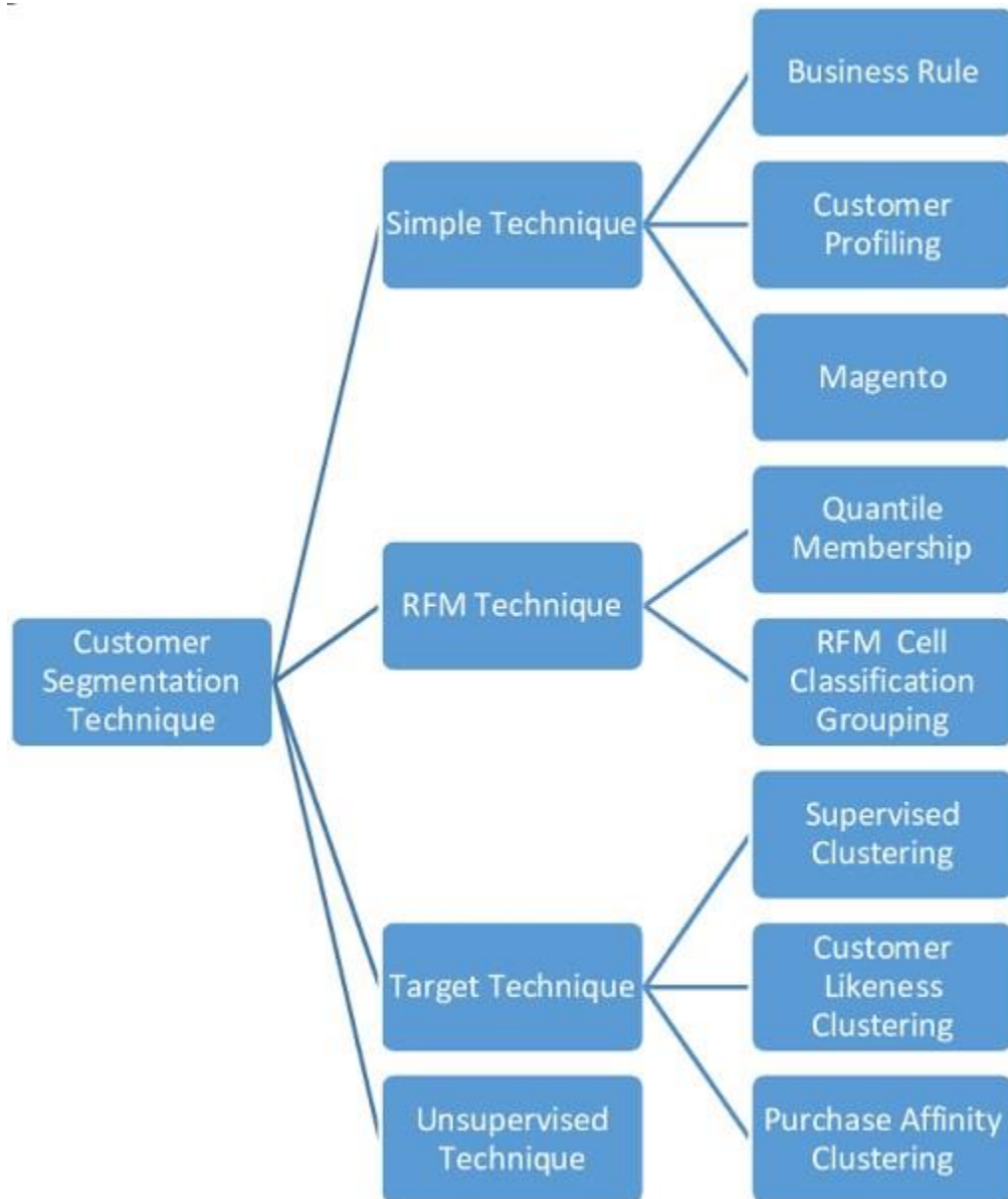
#### **Project Objectives:**

- **Data Collection:** Gather relevant customer data, including demographics, purchase history, and interaction data.
- **Data Preprocessing:** Clean and preprocess the data to ensure it's ready for analysis.
- **Feature Selection/Engineering:** Identify relevant features and possibly create new ones.
- **Algorithm Selection:** Choose the segmentation method or algorithm (e.g., k-means, RFM analysis, or machine learning).
- **Model Training:** Train the chosen algorithm on the data.
- **Segmentation:** Apply the model to segment customers into groups.
- **Evaluation:** Assess the quality of segments using appropriate metrics.
- **Interpretation:** Understand the characteristics of each segment and label them based on customer behavior.
- **Implementation:** Use the segments to tailor marketing strategies and product offerings.
- **Monitoring and Iteration:** Continuously monitor and update segments as needed.

Flow Chart:



**Entity Relationship(ER Diagram):**



### Algorithmic Steps:

- **Clustering Algorithms:** If using clustering, steps might include initializing centroids, assigning data points to clusters, and updating centroids iteratively.
- **RFM Analysis:** Calculate Recency, Frequency, and Monetary scores for each customer and group them accordingly.
- **Machine Learning:** If using machine learning, steps include data splitting, model training, hyperparameter tuning, and prediction.

## PREPROCESSING DATASET:

Here is a general guideline on how to preprocess your dataset for customer segmentation:

1. **Import Libraries:** Begin by importing essential Python libraries for data analysis such as pandas, numpy, and matplotlib or seaborn for visualization.

### Python code:

```
import pandas as pd
```

```
import numpy as np
```

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

2. **Load the Data:** Load your dataset into a pandas DataFrame.

**Python code :**

```
data = pd.read_csv('your_dataset.csv')
```

- 3. Exploratory Data Analysis (EDA):** Perform initial EDA to understand the data, its structure, and any potential issues.

**python code:**

```
# Display the first few rows of the dataset print(data.head())
```

```
# Check the dimensions of the dataset
```

```
print(data.shape) # Check for missing
```

```
values print(data.isnull().sum())
```

```
# Get statistical summary of the dataset print(data.describe)
```

- 4. Data Cleaning:** Handle missing values: Depending on the nature and quantity of missing data, you can choose to drop the rows, fill in the missing values with mean or median, or use advanced imputation techniques.

- 5. Remove duplicates:** Check for and remove any duplicate entries in the dataset.

**Python code:**

```
# Drop rows with missing values
```

```
data = data.dropna() # Remove
```

```
duplicates data =
```

```
data.drop_duplicates()
```

- 6. Data Transformation:**



- Convert categorical variables to numerical ones using encoding techniques such as one-hot encoding or label encoding.
- Scale numerical features if they are on different scales using techniques such as MinMax scaling or Standard scaling.

**Python code:**

```
# Perform one-hot encoding for categorical variables data =
```

```
pd.get_dummies(data, columns=['categorical_column'])
```

```
# Scale numerical features from
```

```
sklearn.preprocessing import MinMaxScaler scaler =
```

```
MinMaxScaler()
```

```
data[['numerical_column1', 'numerical_column2']] = scaler.fit_transform(data[['numerical_column1', 'numerical_column2']])
```

**7. Feature Selection:** Select relevant features that are important for customer segmentation. This can be based on domain knowledge or using feature selection techniques like correlation analysis or using feature importance from machine learning models.

**8. Dimensionality Reduction:** If your dataset has high dimensionality, consider using techniques like Principal Component Analysis (PCA) to reduce the number of features while retaining the most important information.

**Python code:**

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=2) data_pca =
```

```
pca.fit_transform(data)
```

**9. Data Normalization and Standardization:** Ensure that the data is in a suitable format for your chosen machine learning algorithms.

**10. Save Preprocessed Data:** Save the preprocessed data for use in your customer segmentation analysis.

**Python code:**

```
data.to_csv('preprocessed_data.csv', index=False)
```

**Importance of loading and preprocessing dataset :**

Loading and preprocessing the dataset are essential steps in customer segmentation using data science. They play a critical role in ensuring the quality and reliability of the subsequent analysis. Here are the key reasons why these steps are crucial:

**Data Integrity and Quality:** Loading the dataset allows you to check its integrity and assess the quality of the data. By preprocessing the data, you can clean it by handling missing values, removing duplicates, and dealing with outliers. Ensuring data integrity is crucial for obtaining accurate insights from the analysis.

**Data Understanding:** Loading the dataset enables you to understand the structure and format of the data. This understanding is crucial for choosing the appropriate preprocessing techniques. It allows you to identify the types of variables, such as categorical or numerical, and plan the preprocessing steps accordingly.

- 1. Enhanced Analysis Efficiency:** Proper preprocessing enhances the efficiency of the subsequent analysis. It helps in reducing the computational complexity, making the algorithms run faster and enabling the exploration of more sophisticated and complex models without compromising performance.
- 2. Improved Model Performance and Interpretability:** A well-preprocessed dataset leads to improved model performance and interpretability. It ensures that the segmentation model can effectively identify meaningful patterns and segments within the customer data, providing valuable insights for business decisions and marketing strategies.

**Loading the dataset:**

Here are the key steps to load a dataset for customer segmentation:

**1. Data Collection:**

- Identify the sources of customer data, which could include databases, CRM systems, spreadsheets, or external APIs.
- Ensure that you have the necessary permissions and access to retrieve the data.

**2. Data Extraction:**

- Extract the data from the chosen sources. This may involve running SQL queries, using data connectors, or downloading files.

**3. Data Storage:**

- Choose a suitable storage format for your data, such as a CSV file, a relational database, or a data warehouse.
- 
- Consider using data management tools like Pandas (Python) or libraries like SQLAlchemy for efficient data storage.

**4. Data Cleaning:**

- Examine the dataset for missing values, duplicate records, and inconsistencies.
- Perform data cleaning tasks like imputing missing values or removing outliers.

**5. Data Exploration:**

- Explore the loaded dataset to gain an initial understanding of its structure and contents. Use summary statistics and data visualization to identify patterns.

**6. Data Preprocessing:**

- Transform and prepare the data for analysis. This may include encoding categorical variables, scaling numerical features, and feature engineering.

**7. Data Sampling:**

- Depending on the dataset's size, you may choose to take a random sample for initial analysis to save processing time.

**8. Data Splitting:**

- Divide the dataset into training and testing sets if you plan to build and evaluate segmentation models.

#### **9. Data Version Control:**

- If working in a collaborative or version-controlled environment, ensure that you track changes to the dataset to maintain data integrity.

#### **10. Data Security:**

- Take measures to protect the privacy and security of customer data, particularly if it contains sensitive information.

#### **11.Documentation:**

- Maintain clear documentation about the dataset, including its source, structure, and any preprocessing steps applied. This is essential for reproducibility and collaboration.

#### **12.Data Integration:**

- If your analysis requires data from multiple sources, integrate and join datasets as needed.

### **Preprocessing the dataset:**

Preprocessing the dataset is a crucial step in customer segmentation using data science. This phase involves cleaning and transforming the data to make it suitable for analysis. Here are the key steps for preprocessing a dataset in customer segmentation:

#### **1. Data Cleaning:**

- Handle missing values: Decide whether to impute missing data or remove records with missing values based on the impact on the analysis.
- Detect and handle duplicates: Identify and remove duplicate records to prevent bias in segmentation results.

#### **2. Data Transformation:**

- Encoding Categorical Variables: Convert categorical features into a numerical format using techniques like one-hot encoding or label encoding.
- Scaling Numerical Features: Normalize numerical features to ensure that all features have the same scale. Common methods include Min-Max scaling or standardization (mean and standard deviation scaling).

#### **3. Feature Engineering:**

- Create new features: Generate meaningful new features that may better represent customer characteristics.
- Reduce dimensionality: Use dimensionality reduction techniques, such as Principal Component Analysis (PCA), if dealing with high-dimensional data.

#### **4. Handling Outliers:**

- Identify and manage outliers, as they can skew segmentation results. You can use statistical methods or visualization techniques to detect outliers and decide whether to remove or transform them.

#### **5. Data Sampling:**

- If your dataset is large, consider taking a random sample for faster model development and testing. Ensure that the sample is representative of the overall dataset.

#### **6. Data Splitting:**

- Divide the dataset into training and testing sets if you plan to build and evaluate segmentation models. The training set is used for model development, and the testing set is for model evaluation.

#### **7. Data Standardization:**

- Standardize the data if you plan to use clustering techniques that are sensitive to feature scales. Standardization ensures that features have a mean of 0 and a standard deviation of 1.

#### **8. Handling Imbalanced Data:**

- If your dataset has imbalanced segments, consider oversampling or undersampling techniques to balance the representation of segments.

#### **9. Data Imputation:**

- Impute missing data using methods like mean imputation, median imputation, or more advanced imputation techniques.

#### **10. Data Validation:**

- Validate the quality of the preprocessed data to ensure that it aligns with the analysis goals and does not introduce bias or errors.

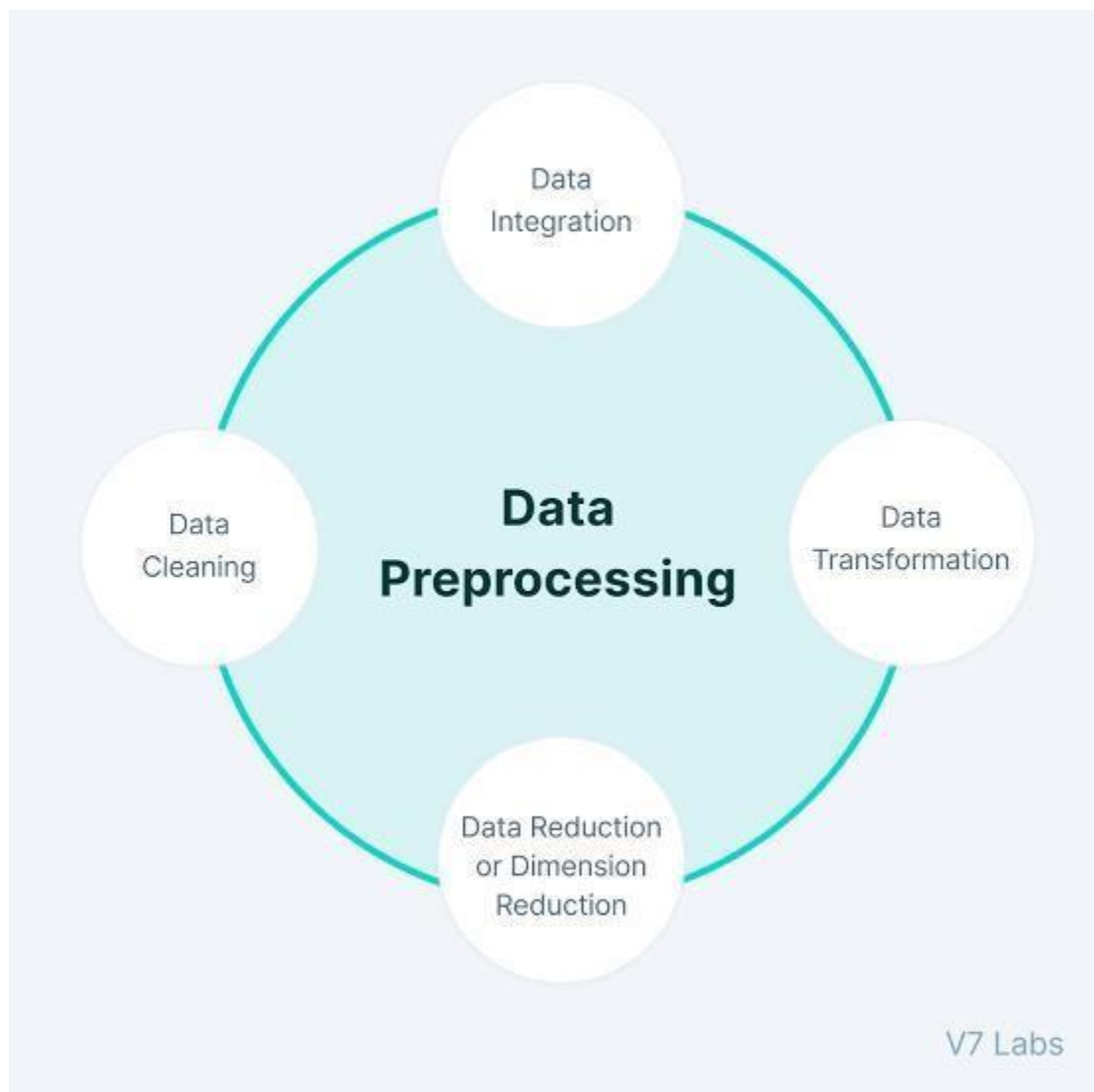
#### **11. Data Version Control:**

- Keep track of changes made to the dataset, especially in collaborative or versioncontrolled environments.

## 12. Documentation:

- Maintain documentation that outlines the preprocessing steps, transformation techniques, and reasons for decisions made during the process. This is essential for reproducibility.

Effective preprocessing of the dataset is critical for accurate and meaningful customer segmentation using data science. It ensures that the data is in a form that can be readily used for clustering or classification algorithms to identify distinct customer segments.



**Import Libraries:**

The libraries we will be required are :

- Pandas
- Numpy
- Matplotlib / Seaborn
- Sklearn

Here, how to load a dataset using Python

**Program:** import

numpy as np import

pandas as pd import

matplotlib.pyplot as plt

import seaborn as sb

from sklearn.preprocessing import StandardScaler, LabelEncoder from

sklearn.cluster import KMeans

import warnings warnings.filterwarnings('ignore')

**Importing Dataset:**

The dataset taken for the task includes the details of customers includes their marital status, their income, number of items purchased, types of items purchased, and so on.

**Program:** df =

pd.read\_csv('new.csv')

df.head()

Output:

```
ID Year_Birth Education Marital_Status Income Kidhome Teenhome \
0 5524 1957 Graduation Single 58138.0 0 0
1 2174 1954 Graduation Single 46344.0 1 1
2 4141 1965 Graduation Together 71613.0 0 0
3 6182 1984 Graduation Together 26646.0 1 0
4 5324 1981 PhD Married 58293.0 1 0

Dt_Customer Recency MntWines ... NumDealsPurchases NumWebPurchases \
0 04-09-2012 58 635 ... 3 8
1 08-03-2014 38 11 ... 2 1
2 21-08-2013 26 426 ... 1 8
3 10-02-2014 26 11 ... 2 2
4 19-01-2014 94 173 ... 5 5

NumCatalogPurchases NumStorePurchases NumWebVisitsMonth Complain \
0 10 4 7 0
1 1 2 5 0
2 2 10 4 0
3 0 4 6 0
4 3 6 5 0

Z_CostContact Z_Revenue Response Accepted
0 3 11 1 AcceptedCmp1
1 3 11 0 AcceptedCmp1
2 3 11 0 AcceptedCmp1
3 3 11 0 AcceptedCmp1
4 3 11 0 AcceptedCmp1

[5 rows x 25 columns]
```

To check the shape of the dataset we can use data.shape method.

Program: df.shape

Output:

```
(2240, 25)(2240, 25)
```

To get the information of the dataset like checking the null values, count of values, etc. we will use .info() method. **Data Processing**

Program:

```
df.info()
```

Output:



```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 25 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                -
0   ID                                    2240 non-null   int64
1   Year_Birth                           2240 non-null   int64
2   Education                             2240 non-null   object
3   Marital_Status                       2240 non-null   object
4   Income                               2216 non-null   float64
5   Kidhome                              2240 non-null   int64
6   Teenhome                             2240 non-null   int64
7   Dt_Customer                          2240 non-null   object
8   Recency                              2240 non-null   int64
9   MntWines                             2240 non-null   int64
10  MntFruits                            2240 non-null   int64
11  MntMeatProducts                      2240 non-null   int64
12  MntFishProducts                      2240 non-null   int64
13  MntSweetProducts                    2240 non-null   int64
14  MntGoldProds                        2240 non-null   int64
15  NumDealsPurchases                   2240 non-null   int64
16  NumWebPurchases                     2240 non-null   int64
17  NumCatalogPurchases                 2240 non-null   int64
18  NumStorePurchases                   2240 non-null   int64
19  NumWebVisitsMonth                   2240 non-null   int64
20  Complain                             2240 non-null   int64
21  Z_CostContact                       2240 non-null   int64
22  Z_Revenue                           2240 non-null   int64
23  Response                             2240 non-null   int64
24  Accepted                             2240 non-null   object
dtypes: float64(1), int64(20), object(4)

```

**Program:** df.describe().T

**Output:**

	count	mean	std	min	25%	50%	75%	max
ID	2240.0	5592.159821	3246.662198	0.0	2828.25	5458.5	8427.75	11191.0
Year_Birth	2240.0	1968.805804	11.984069	1893.0	1959.00	1970.0	1977.00	1996.0
Income	2216.0	52247.251354	25173.076861	1730.0	35303.00	51381.5	68522.00	666666.0
Kidhome	2240.0	0.444196	0.538398	0.0	0.00	0.0	1.00	2.0
Teenhome	2240.0	0.506250	0.544538	0.0	0.00	0.0	1.00	2.0
Recency	2240.0	49.109375	28.962453	0.0	24.00	49.0	74.00	99.0
MntWines	2240.0	303.935714	336.597393	0.0	23.75	173.5	504.25	1493.0
MntFruits	2240.0	26.302232	39.773434	0.0	1.00	8.0	33.00	189.0
MntMeatProducts	2240.0	166.950000	225.715373	0.0	16.00	67.0	232.00	1725.0
MntFishProducts	2240.0	37.525446	54.628979	0.0	3.00	12.0	50.00	259.0
MntSweetProducts	2240.0	27.062946	41.280498	0.0	1.00	8.0	33.00	263.0
MntGoldProds	2240.0	44.021875	52.167439	0.0	9.00	24.0	56.00	362.0
NumDealsPurchases	2240.0	2.325000	1.932238	0.0	1.00	2.0	3.00	15.0
NumWebPurchases	2240.0	4.084821	2.778714	0.0	2.00	4.0	6.00	27.0
NumCatalogPurchases	2240.0	2.662054	2.923101	0.0	0.00	2.0	4.00	28.0
NumStorePurchases	2240.0	5.790179	3.250958	0.0	3.00	5.0	8.00	13.0
NumWebVisitsMonth	2240.0	5.316518	2.426645	0.0	3.00	6.0	7.00	20.0
Complain	2240.0	0.009375	0.096391	0.0	0.00	0.0	0.00	1.0
Z_CostContact	2240.0	3.000000	0.000000	3.0	3.00	3.0	3.00	3.0
Z_Revenue	2240.0	11.000000	0.000000	11.0	11.00	11.0	11.00	11.0
Response	2240.0	0.149107	0.356274	0.0	0.00	0.0	0.00	1.0

Improving the values in the Accepted column.

**Program:**

```
df['Accepted'] = df['Accepted'].str.replace('Accepted', '')
```

To check the null values in the dataset

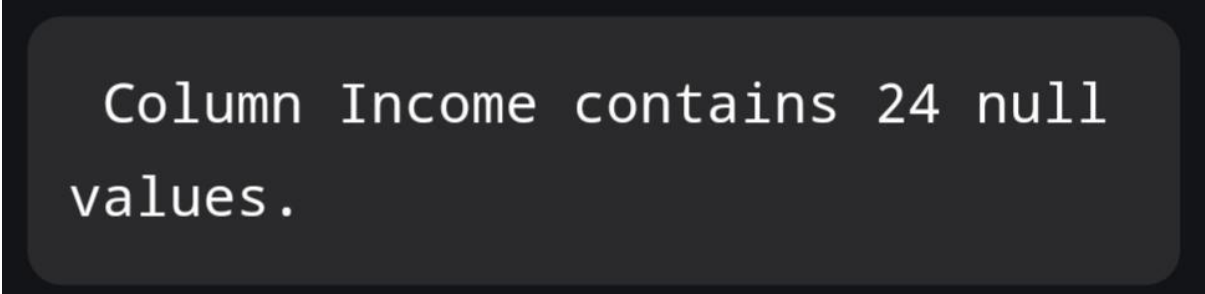
**Program:**

```
for col in df.columns:
```

```
    temp = df[col].isnull().sum()
```

```
    if temp > 0:
```

```
        print(f'Column {col} contains {temp} null values.') Output:
```



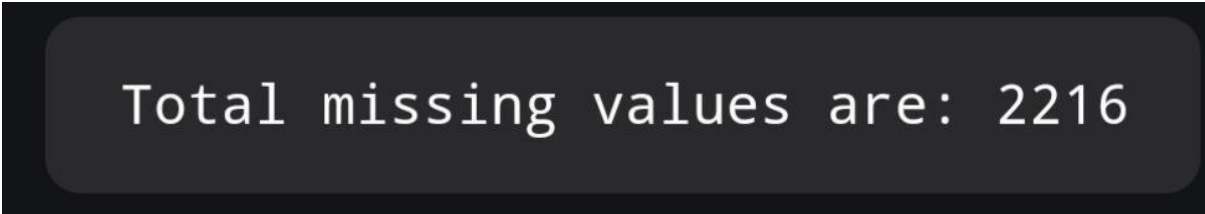
```
Column Income contains 24 null
values.
```

Now, once we have the count of the null values and we know the values are very less we can drop them (it will not affect the dataset much).

**Program:**

```
df = df.dropna() print("Total missing values
are:", len(df))
```

**Output :**



```
Total missing values are: 2216
```

To find the total number of unique values in each column we can use data.unique() method.

**Program:** df.nunique()

**Output:**

---

ID	2216
Year_Birth	59
Education	5
Marital_Status	8
Income	1974
Kidhome	3
Teenhome	3
Dt_Customer	662
Recency	100
MntWines	776
MntFruits	158
MntMeatProducts	554
MntFishProducts	182
MntSweetProducts	176
MntGoldProds	212
NumDealsPurchases	15
NumWebPurchases	15
NumCatalogPurchases	14
NumStorePurchases	14
NumWebVisitsMonth	16
Complain	2
Z_CostContact	1
Z_Revenue	1
Response	2
Accepted	5

Here we can observe that there are columns which contain single values in the whole column so, they have no relevance in the model development.

Also dataset has a column Dt\_Customer which contains the date column, we can convert into 3 columns i.e. day, month, year.

**Program:**

```
parts = df["Dt_Customer"].str.split("-", n=3, expand=True)

df["day"] = parts[0].astype('int') df["month"] =

parts[1].astype('int') df["year"] = parts[2].astype('int')
```

Now we have all the important features, we can now drop features like Z\_CostContact, Z\_Revenue, Dt\_Customer.

**Program:**

```
df.drop(['Z_CostContact', 'Z_Revenue', 'Dt_Customer'],  
        axis=1,  
        inplace=True)
```

**Data Visualization and Analysis**

Data Visualization is the graphical representation of information and data in a pictorial or graphical format. Here we will be using bar plot and count plot for better visualization.

**Program:**

```
floats, objects = [], []  
for col in df.columns:  
    if df[col].dtype == object:  
        objects.append(col)  
    elif df[col].dtype == float:  
        floats.append(col)  
  
print(objects) print(floats)
```

**Output:**

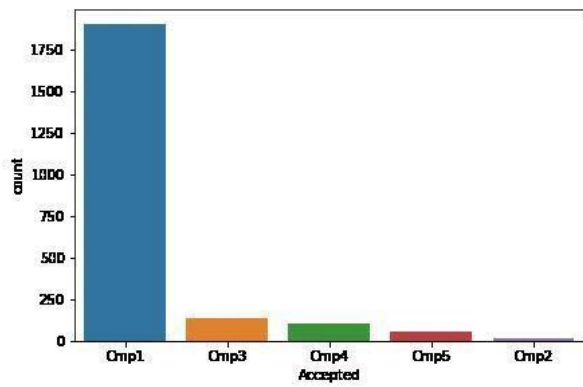
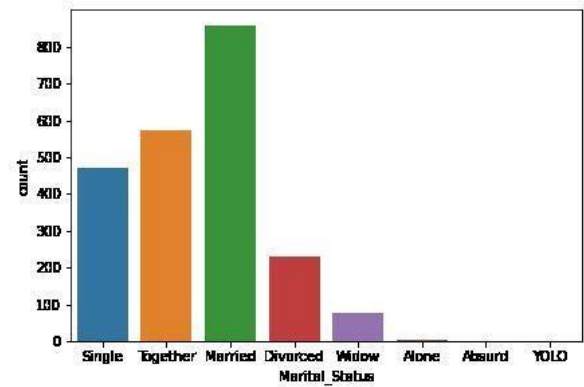
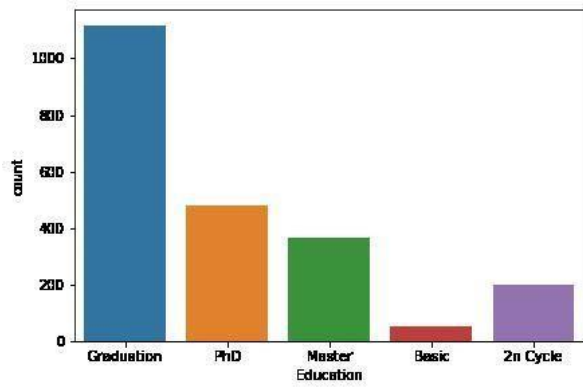
```
['Education', 'Marital_Status',  
'Accepted']  
['Income']
```

To get the count plot for the columns of the datatype – object, refer the code below.

**Program:**

```
plt.subplots(figsize=(15, 10)) for i,  
col in enumerate(objects):  
  
plt.subplot(2, 2, i + 1)  
  
sb.countplot(df[col])  
  
plt.show()
```

**Output:**



Let's check the value\_counts of the Marital\_Status of the data.

**Prrogram:**

```
df['Marital_Status'].value_counts()
```

**Output:**

```

Married      857
Together     573
Single       471
Divorced     232
Widow        76
Alone         3
Absurd        2
YOLO         2
Name: Marital_Status, dtype: int64

```

Now lets see the comparison of the features with respect to the values of the responses.

**Program:**

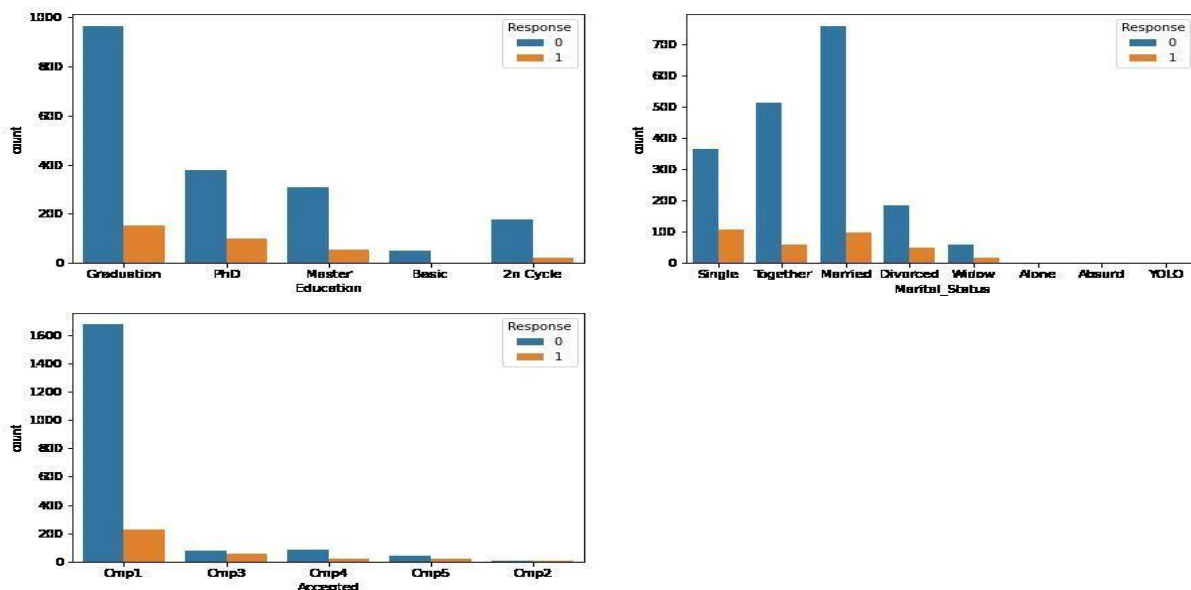
```

plt.subplots(figsize=(15, 10)) for i, col in
enumerate(objects):    plt.subplot(2, 2, i + 1)

sb.countplot(df[col], hue=df['Response']) plt.show()

```

**Output:**



**Label Encoding**



Label Encoding is used to convert the categorical values into the numerical values so that model can understand it.

### Program:

```
for col in df.columns:    if
```

```
df[col].dtype == object:
```

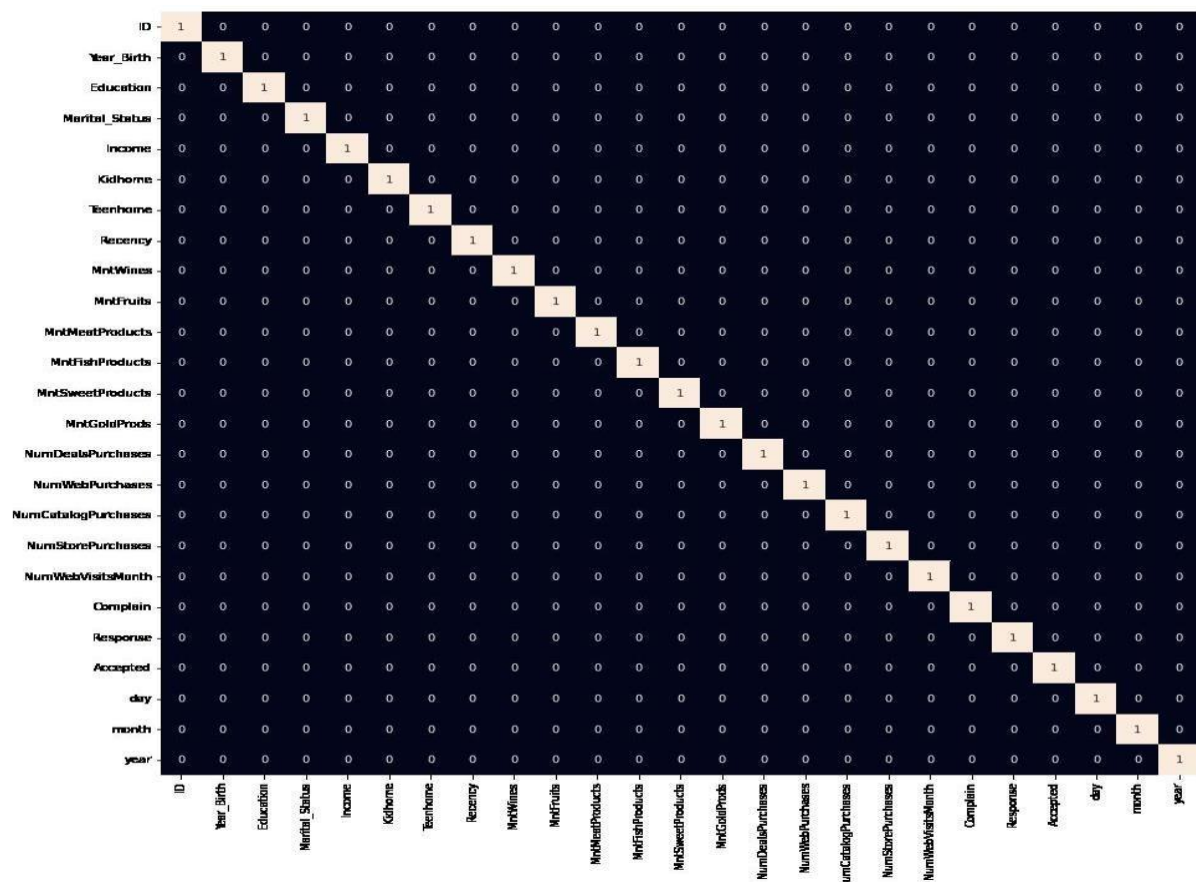
```
    le = LabelEncoder()
```

```
    df[col] = le.fit_transform(df[col])
```

Heat map is the best way to visualize the correlation among the different features of dataset. Let's give it the value of 0.8 **Program :**

```
plt.figure(figsize=(15, 15)) sb.heatmap(df.corr() > 0.8,
```

```
annot=True, cbar=False) plt.show() Output:
```



## Standardization

Standardization is the method of feature scaling which is an integral part of feature engineering. It scales down the data and making it easier for the machine learning model to learn from it. It reduces the mean to '0' and the standard deviation to '1'.

### Program:

```
scaler = StandardScaler() data =  
scaler.fit_transform(df)
```

## Segmentation

We will be using T-distributed Stochastic Neighbor Embedding. It helps in visualizing highdimensional data. It converts similarities between data points to joint probabilities and tries to minimize the values to low-dimensional embedding.

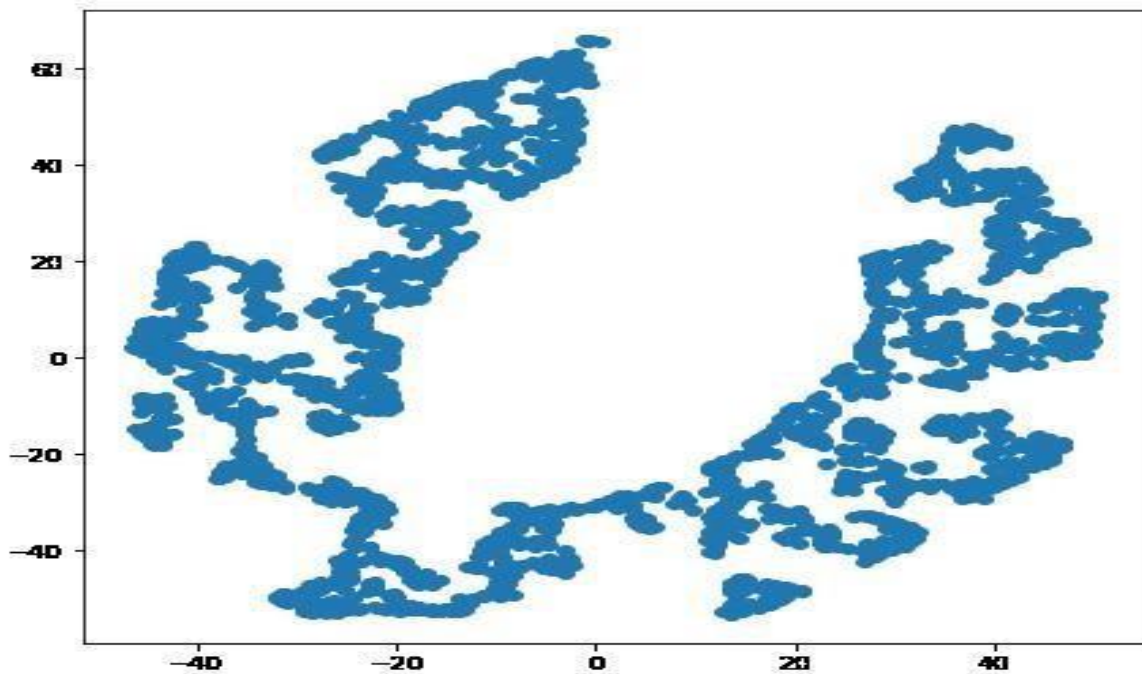
**Program:** from sklearn.manifold import TSNE model =

TSNE(n\_components=2, random\_state=0) tsne\_data =

model.fit\_transform(df) plt.figure(figsize=(7, 7))

plt.scatter(tsne\_data[:, 0], tsne\_data[:, 1]) plt.show()

### Output:



There are certainly some clusters which are clearly visual from the 2-D representation of the given data. Let's use the KMeans algorithm to find those clusters in the high dimensional plane itself

KMeans Clustering can also be used to cluster the different points in a plane.

**Program:**

```
error = [] for n_clusters in
```

```
range(1, 21):
```

```
    model = KMeans(init='k-means++',
```

```
                    n_clusters=n_clusters,
```

```
                    max_iter=500,
```

```
                    random_state=22)
```

```
    model.fit(df)    error.append(model.inertia_)
```



Here inertia is nothing but the sum of squared distances within the cluster.

**Program:**

```
plt.figure(figsize=(10, 5))
sb.lineplot(x=range(1, 21), y=error)
sb.scatterplot(x=range(1, 21), y=error)
plt.show()
```

**Output:**

Here by using the elbow method we can say that  $k = 6$  is the optimal number of clusters that should be made as after  $k = 6$  the value of the inertia is not decreasing drastically.

**Program:**

```
# create clustering model with optimal k=5 model =
KMeans(init='k-means++',
        n_clusters=5,
        max_iter=500, random_state=22)

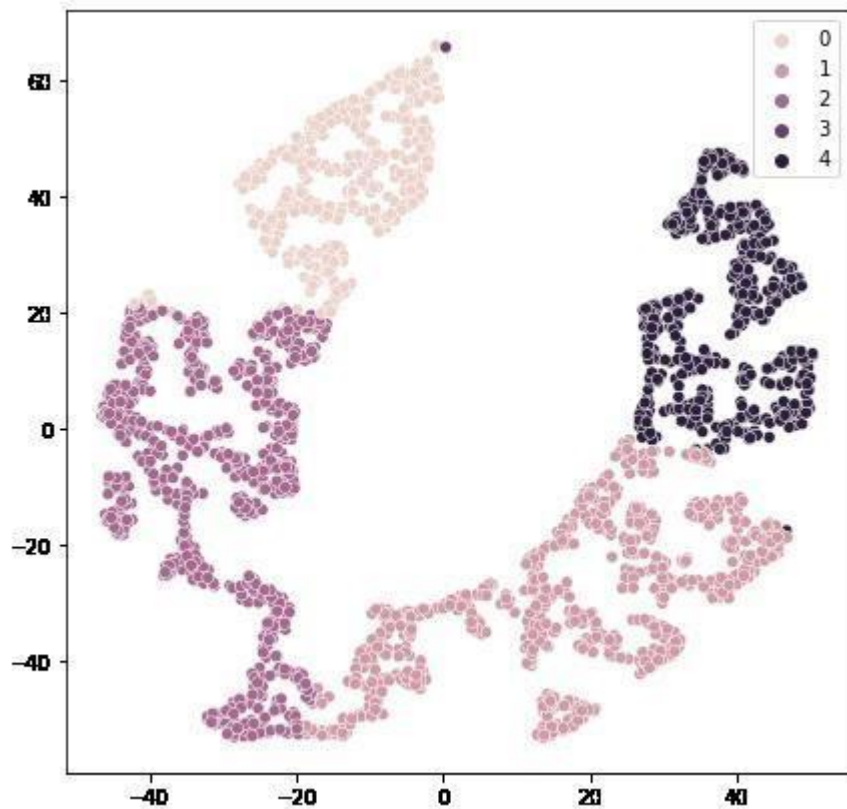
segments = model.fit_predict(df)
```

Scatterplot will be used to see all the 6 clusters formed by KMeans Clustering.

**Program:**

```
plt.figure(figsize=(7, 7)) sb.scatterplot(tsne_data[:, 0], tsne_data[:,
1], hue=segments) plt.show()
```

**Output:**



### Overview of the Process:

The process of customer segmentation using data science generally involves several key steps:

1. **Data Collection:** Gather relevant data about customers, including demographic information, purchase history, interactions, and other relevant variables.
2. **Data Preprocessing:** Clean the data, handle missing values, and standardize or normalize the data to ensure consistency and accuracy.
3. **Feature Selection:** Identify the most relevant features or attributes that are crucial for segmenting customers effectively.

4. **Exploratory Data Analysis (EDA):** Conduct in-depth exploratory analysis to understand the distribution, correlations, and patterns within the data, using techniques such as data visualization and statistical analysis.
5. **Segmentation Technique Selection:** Choose appropriate data science techniques such as clustering algorithms (e.g., k-means, hierarchical clustering), classification models, or collaborative filtering, depending on the nature of the data and the specific business goals.
6. **Segmentation Model Building:** Apply the selected techniques to group customers into distinct segments based on similarities in their behaviors, preferences, or characteristics.
7. **Validation and Evaluation:** Assess the quality and effectiveness of the segmentation by using metrics such as silhouette scores, within-cluster sum of squares, or other domain-specific evaluation criteria.
8. **Interpretation of Segments:** Analyze and interpret the characteristics of each customer segment to gain actionable insights that can inform marketing strategies and business decisions.
9. **Implementation of Marketing Strategies:** Tailor marketing campaigns, product recommendations, and customer interactions based on the unique needs and preferences of each identified customer segment.
10. **Monitoring and Refinement:** Continuously monitor the performance of the segmentation strategy and refine the approach based on feedback, new data, and changes in customer behavior over time.

This iterative process enables businesses to understand their customers more deeply and create targeted marketing approaches that can lead to improved customer satisfaction and higher profitability.

## PROCEDURE:

### Features Explanation:

In a customer segmentation using data science, feature engineering plays a crucial role in preparing the data for effective analysis and segmentation. Here are some key aspects of feature engineering in this context:

1. **Feature Extraction:** Extract meaningful features from the raw data that can help in distinguishing different customer segments. This may involve deriving new variables from existing ones or transforming data into a more suitable format for analysis.

2. **Normalization and Scaling:** Normalize and scale the features to ensure that all variables are on a comparable scale, which is essential for certain algorithms like k-means clustering that are sensitive to the scale of the data.
3. **Encoding Categorical Variables:** Convert categorical variables into a format that can be easily interpreted by machine learning algorithms, such as one-hot encoding or label encoding, enabling the algorithms to process the data effectively.
4. **Handling Missing Data:** Implement strategies to handle missing data points, either through imputation techniques or by removing the data carefully, ensuring that the absence of certain data points does not skew the segmentation results.
5. **Creation of Derived Features:** Create new features based on domain knowledge or hypotheses that might be relevant in distinguishing customer segments. These derived features can be combinations of existing variables or interactions between different attributes.
6. **Dimensionality Reduction:** Employ dimensionality reduction techniques such as principal component analysis (PCA) or t-distributed stochastic neighbor embedding (t-SNE) to reduce the complexity of the dataset while retaining the most crucial information for segmentation.
7. **Temporal Feature Engineering:** Incorporate time-based features, seasonal patterns, or trends that can provide valuable insights into customer behavior changes over different time periods.

By carefully engineering features, data scientists can ensure that the data is optimized for segmentation, allowing for more accurate and meaningful identification of customer segments that can guide targeted marketing strategies and personalized customer experiences.

### **Applying Clustering Algorithms:**

In customer segmentation, various clustering algorithms can be applied to group customers into distinct segments based on their similarities. Some commonly used clustering algorithms for customer segmentation include:

1. **K-means Clustering:** A popular and efficient algorithm that partitions the data into K clusters, minimizing the sum of squared distances between data points and the centroid of the clusters.
2. **Hierarchical Clustering:** Builds a hierarchy of clusters by either recursively merging or splitting clusters based on the distance between data points, creating a dendrogram that visually represents the clusters.

## K-Means Clustering:

### Import Libraries:

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
%matplotlib inline

from IPython.display import Image
```

### Loading Data:

In [2]:

```
df = pd.read_csv('/kaggle/input/mall-customers/Mall_Customers.csv')
df.head()
```

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

### Creating the Clusters:

In [3]:

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

In [4]:

```
wcss = []
```

In [5]:

```
for i in range(1,11):
    km = KMeans(n_clusters=i,init = 'k-means++',max_iter=300,n_init=10,random_state=0)
    km.fit(X)
    wcss.append(km.inertia_)
```

### Applying K-means to mall dataset:

In [6]:

```
km = KMeans(n_clusters=5,init = 'k-means++',max_iter=300,n_init=10,random_state=0) # setting default values for max_iter and n_init
y_means = km.fit_predict(X)
```

### Visualizing the Clusters:

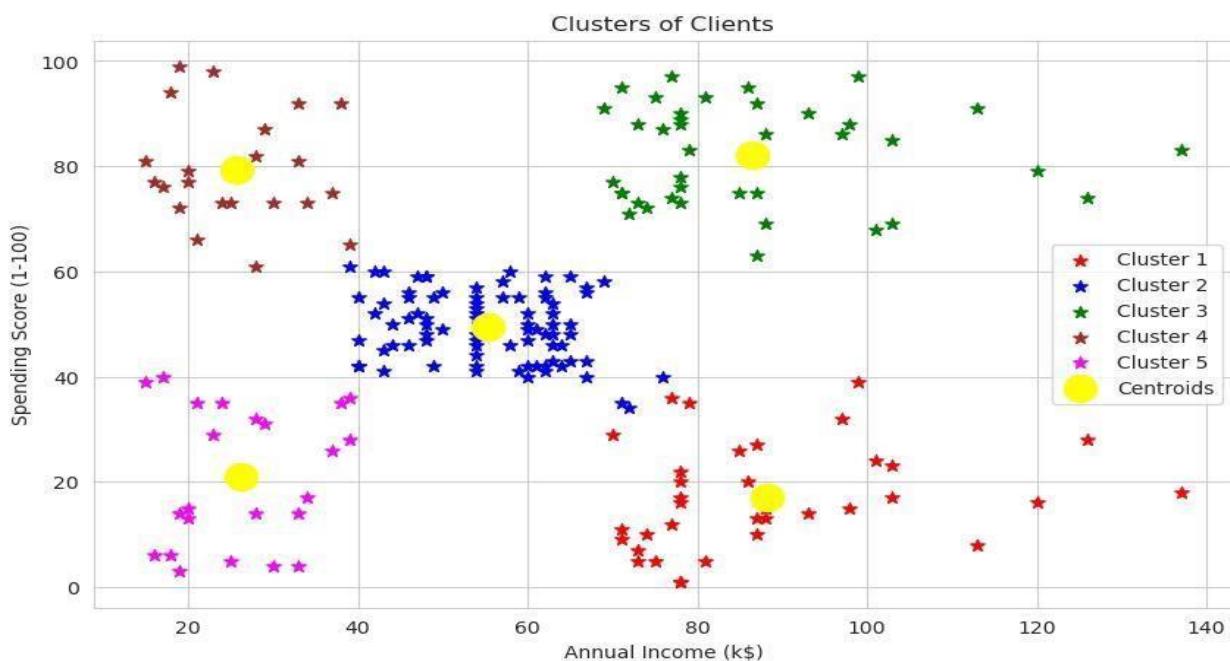
In [7]:

```

#Plotting scatter plot of clusters along with their highlighted
clusters. sns.set_style('whitegrid') plt.figure(figsize=(10,6))
plt.scatter(X[y_means==0,0],X[y_means==0,1],s=50,c='red',label='Cluster 1'
,marker='*') #X[y_means==0,0] for x-coordinates for cluter1,X[y_means==0,1]
for y-coordinates ,s for size of datapoint
plt.scatter(X[y_means==1,0],X[y_means==1,1],s=50,c='blue',label='Cluster 2'
,marker='*')
plt.scatter(X[y_means==2,0],X[y_means==2,1],s=50,c='green',label='Cluster
3',marker='*')
plt.scatter(X[y_means==3,0],X[y_means==3,1],s=50,c='brown',label='Cluster
4',marker='*')
plt.scatter(X[y_means==4,0],X[y_means==4,1],s=50,c='magenta',label='Cluste
r 5',marker='*')
plt.scatter(km.cluster_centers[:,0],km.cluster_centers[:,1],s=250,c='yel
low',label='Centroids') #Centroids are highlighted with bigger size

plt.title('Clusters of Clients')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend();

```



- **Cluster 1** - In this cluster, clients earn High income but don't spend much money. We could call these clients as "Careful"

- **Cluster 2** - In this cluster, clients earn average income and have average spending score. Let's call them "Standard"
- **Cluster 3** - In this cluster, clients earn High income and have high spending score. This is main potential customer base for mall's marketing campaigns and it would be very insightful for the mall to understand what kind of product are brought by these set of clients. So we would call this cluster as "Target"
- **Cluster 4** - In this cluster, clients earn low income but high spending score. Let's call these clients "Careless"
- **Cluster 5** - In this cluster, clients earn low income and have low spending score. Let's name them "Sensible"

## Hierarchical Clustering:

In[1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
# Input data files are available in the read-only "../input/" directory #
# For example, running this (by clicking run or pressing Shift+Enter) will
# list all files under the input directory
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

kaggle/input/mall-customers/Mall\_Customers.csv

## Import Libraries:

In [2]:

```
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
import matplotlib.pyplot as plt
```

In [3]:

```
#Reading Data
```

```
train_path = "/kaggle/input/mall-  
customers/Mall_Customers.csv" train_data =  
pd.read_csv(train_path)
```

In [4]:

```
#Checking the data train_data.head()
```

Out[4]:

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

In [5]:

```
#Choosing potential features for clustering such as Annual Income and Spen  
ding Score  
train_data = train_data.drop(columns=["CustomerID", "Genre", "Age"], axis=1)
```

In [6]:

```
#Checking the data after dropping columns  
train_data.head()
```

Out[6]:

	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

In [7]:

```
#Standardizing the dataset sc
= StandardScaler()
train_data = sc.fit_transform(train_data)
train_data
```

Out[7]:

```
array([[ -1.73899919, -0.43480148],          [-
1.73899919,  1.19570407],
       [-1.70082976, -1.71591298],
       [-1.70082976,  1.04041783],
       [-1.66266033, -0.39597992],
       [-1.66266033,  1.00159627],
       [-1.62449091, -1.71591298],
       [-1.62449091,  1.70038436],
       [-1.58632148, -1.83237767],
       [-1.58632148,  0.84631002],
       [-1.58632148, -1.4053405 ],
       [-1.58632148,  1.89449216],
       [-1.54815205, -1.36651894],
       [-1.54815205,  1.04041783],
       [-1.54815205, -1.44416206],
       [-1.54815205,  1.11806095],
       [-1.50998262, -0.59008772],
       [-1.50998262,  0.61338066],
       [-1.43364376, -0.82301709],
       [-1.43364376,  1.8556706 ],
       [-1.39547433, -0.59008772],
       [-1.39547433,  0.88513158],
```

[-1.3573049 , -1.75473454],  
[-1.3573049 , 0.88513158],  
[-1.24279661, -1.4053405 ],  
[-1.24279661, 1.23452563],  
[-1.24279661, -0.7065524 ],  
[-1.24279661, 0.41927286],  
[-1.20462718, -  
0.74537397],  
[-1.20462718, 1.42863343],  
[-1.16645776, -1.7935561 ],  
[-1.16645776, 0.88513158],  
[-1.05194947, -1.7935561 ],  
[-1.05194947, 1.62274124],  
[-1.05194947, -1.4053405 ],  
[-1.05194947, 1.19570407],  
[-1.01378004, -1.28887582],  
[-1.01378004, 0.88513158],  
[-0.89927175, -0.93948177],  
[-0.89927175, 0.96277471],  
[-0.86110232, -0.59008772],  
[-0.86110232, 1.62274124],  
[-0.82293289, -0.55126616],  
[-0.82293289, 0.41927286],  
[-0.82293289, -0.86183865],  
[-0.82293289, 0.5745591 ],  
[-0.78476346, 0.18634349],  
[-0.78476346, -0.12422899],  
[-0.78476346, -0.3183368 ],  
[-0.78476346, -0.3183368 ],  
[-0.70842461, 0.06987881],  
[-0.70842461, 0.38045129],  
[-0.67025518, 0.14752193],  
[-0.67025518, 0.38045129],  
[-0.67025518, -0.20187212],  
[-0.67025518, -0.35715836],  
[-0.63208575, -0.00776431],  
[-0.63208575, -0.16305055],  
[-0.55574689, 0.03105725],  
[-0.55574689, -0.16305055],  
[-0.55574689, 0.22516505],  
[-0.55574689,  
0.18634349],

[-0.51757746,  
0.06987881],  
[-0.51757746,  
0.34162973],  
[-0.47940803,  
0.03105725],  
[-0.47940803,  
0.34162973], [-0.47940803,  
-0.00776431],  
[-0.47940803, -  
0.08540743], [-0.47940803,  
0.34162973], [-0.47940803,  
-0.12422899], [-0.4412386  
, 0.18634349], [-  
0.4412386, -0.3183368 ],  
[-0.40306917, -  
0.04658587], [-0.40306917,  
0.22516505],  
[-0.25039146, -  
0.12422899],  
[-0.25039146, 0.14752193],  
[-0.25039146, 0.10870037],  
[-0.25039146, -0.08540743],  
[-0.25039146, 0.06987881],  
[-0.25039146, -0.3183368 ],  
[-0.25039146, 0.03105725],  
[-0.25039146, 0.18634349],  
[-0.25039146, -0.35715836],  
[-0.25039146, -0.24069368],  
[-0.25039146, 0.26398661],  
[-0.25039146, -0.16305055],  
[-0.13588317, 0.30280817],  
[-0.13588317, 0.18634349],  
[-0.09771374, 0.38045129],  
[-0.09771374, -0.16305055],  
[-0.05954431, 0.18634349],  
[-0.05954431, -0.35715836],  
[-0.02137488, -0.04658587],  
[-0.02137488, -0.39597992],  
[-0.02137488, -0.3183368 ],  
[-0.02137488, 0.06987881],  
[-0.02137488, -0.12422899],

[-0.02137488, -0.00776431],  
[ 0.01679455, -0.3183368 ],  
[ 0.01679455, -0.04658587],  
[ 0.05496398, -0.35715836],  
[ 0.05496398, -0.08540743],  
[ 0.05496398, 0.34162973],  
[ 0.05496398, 0.18634349],  
[ 0.05496398, 0.22516505],  
[ 0.05496398, -0.3183368 ],  
[ 0.09313341, -0.00776431],  
[ 0.09313341, -  
0.16305055],  
[ 0.09313341, -  
0.27951524],  
[ 0.09313341, -  
0.08540743], [ 0.09313341,  
0.06987881],  
[ 0.09313341,  
0.14752193], [ 0.13130284,  
-0.3183368 ],  
[ 0.13130284, -  
0.16305055],  
[ 0.16947227, -  
0.08540743],  
[ 0.16947227, -  
0.00776431],  
[ 0.16947227, -0.27951524],  
[ 0.16947227, 0.34162973],  
[ 0.24581112, -0.27951524],  
[ 0.24581112, 0.26398661],  
[ 0.24581112,  
0.22516505],  
[ 0.24581112, -0.39597992],  
[ 0.32214998, 0.30280817],  
[ 0.32214998, 1.58391968],  
[ 0.36031941, -0.82301709],  
[ 0.36031941, 1.04041783],  
[ 0.39848884, -0.59008772],  
[ 0.39848884, 1.73920592],  
[ 0.39848884, -1.52180518],  
[ 0.39848884, 0.96277471],  
[ 0.39848884, -1.5994483 ],



[ 0.39848884, 0.96277471],  
[ 0.43665827, -0.62890928],  
[ 0.43665827, 0.80748846],  
[ 0.4748277, -1.75473454],  
[ 0.4748277, 1.46745499],  
[ 0.4748277, -1.67709142],  
[ 0.4748277, 0.88513158],  
[ 0.51299713, -1.56062674],  
[ 0.51299713, 0.84631002],  
[ 0.55116656, -1.75473454],  
[ 0.55116656, 1.6615628 ],  
[ 0.58933599, -0.39597992],  
[ 0.58933599, 1.42863343],  
[ 0.62750542, -1.48298362],  
[ 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],

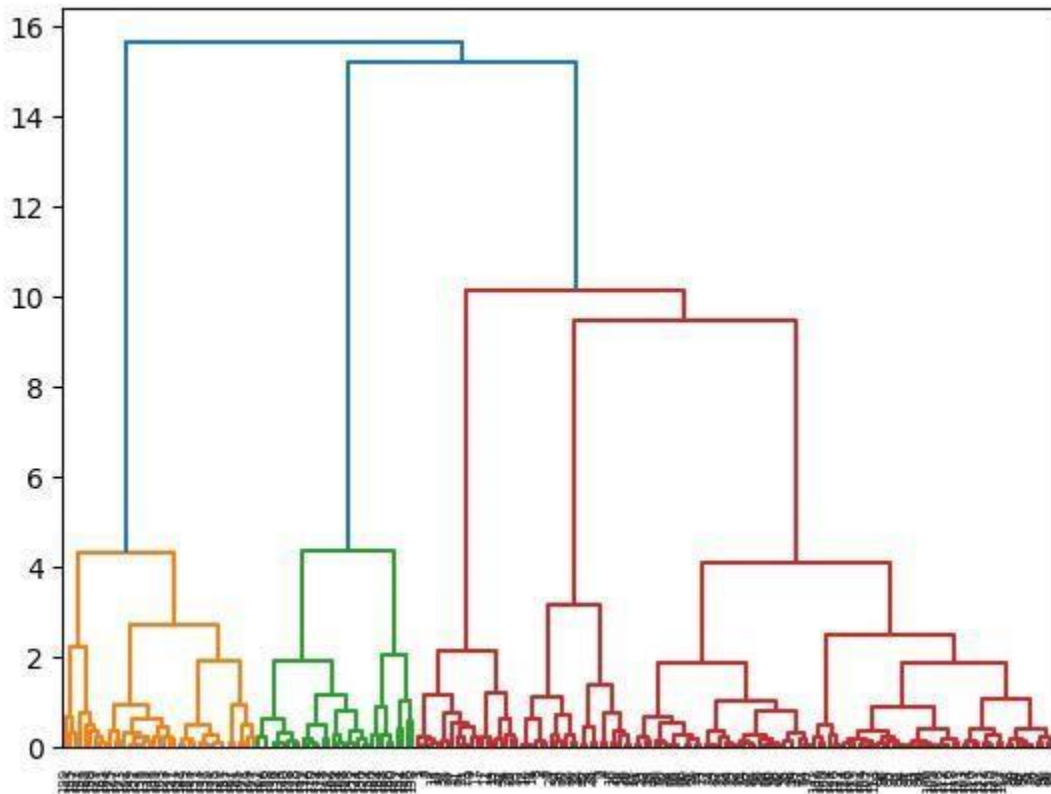
```

[ 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]])

```

In [8]:

```
#Dendrogram
linkage_data = linkage(train_data, method='ward',
metric='euclidean') dendrogram(linkage_data)
plt.show()
```



In [9]:

```
#Hierarchial Clustering Model
hierarchical_cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
labels = hierarchical_cluster.fit_predict(train_data)
```

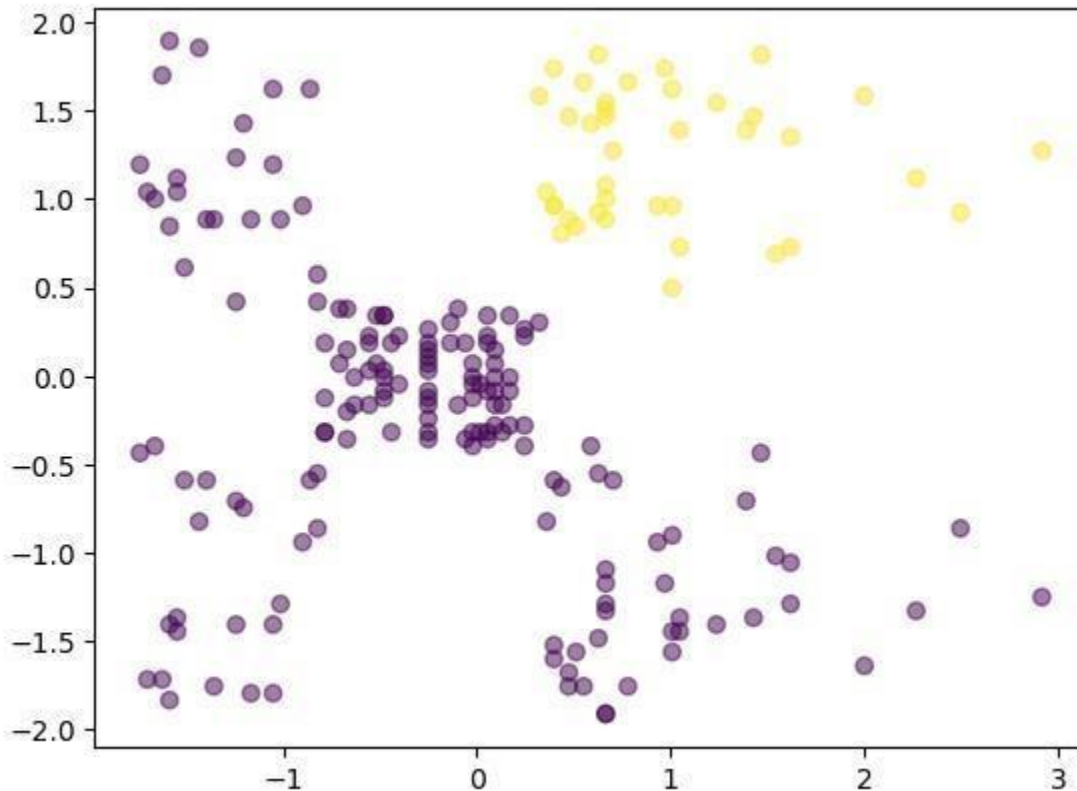
### Visualization of the Clustering:

In [10]:

```
#Visualizing the clusters of Hierarchial Clustering xs =
train_data[:,0] ys = train_data[:,1]
plt.scatter(xs,ys,c=labels,alpha=0.5)
```

Out[10]:

<matplotlib.collections.PathCollection at 0x799566dd0990>



### Customer Segmentation:

- This Python 3 environment comes with many helpful analytics libraries installed
- It is defined by the kaggle/python Docker image: <https://github.com/kaggle/docker-python>
- For example, here's several helpful packages to load.

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
# Input data files are available in the read-only "../input/" directory #
# For example, running this (by clicking run or pressing Shift+Enter) will
# list all files under the input directory
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

/kaggle/input/mall-customers/Mall\_Customers.csv

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
df=pd.read_csv('/kaggle/input/mall-customers/Mall_Customers.csv')

df.rename(columns={'Genre':'Gender'},inplace=True)
df.head()
```

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

In [4]:

```
df.describe()
```

Out[4]:

	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 [5]:

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

Out[5]:

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

In [6]:

```
df.drop(['CustomerID'],axis=1,inplace=True)
```

In [7]:

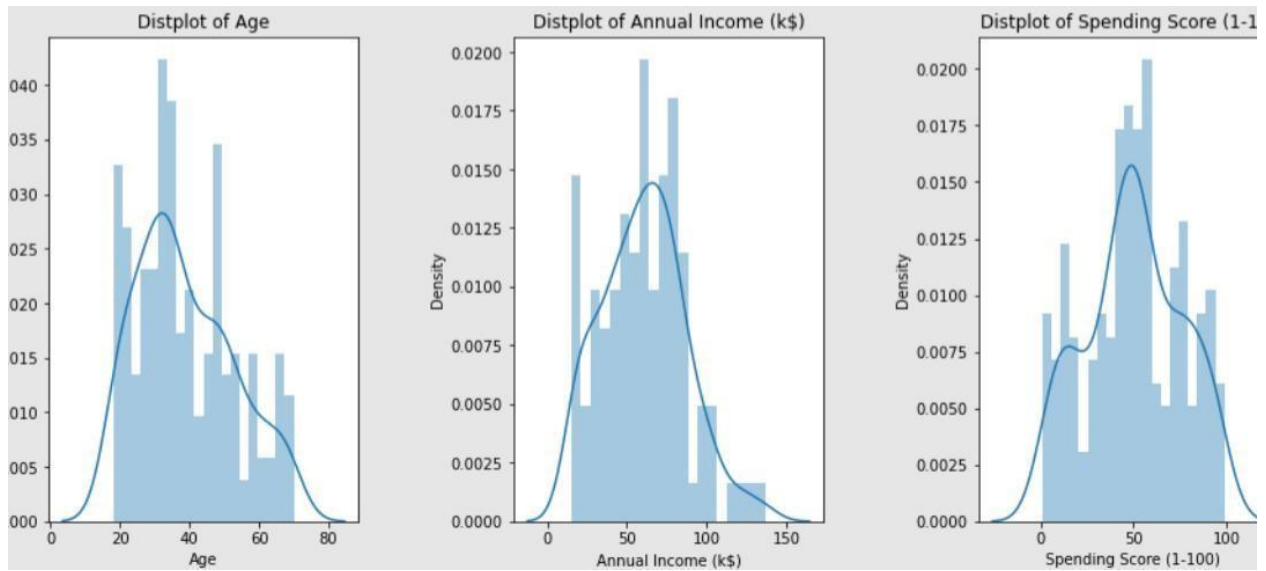
```
plt.figure(1,figsize=(15,6)) n = 0 for x in ['Age','Annual
Income (k$)','Spending Score (1-100)']:    n +=1
    plt.subplot(1,3,n)
    plt.subplots_adjust(hspace=0.5,wspace=0.5)
sns.distplot(df[x],bins=20)    plt.title('Distplot
of {}'.format(x)) plt.show()
```

/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557:  
FutureWarning: `distplot` is a deprecated function and will be removed  
in a future version. Please adapt your code to use either `displot` (a  
figure-level function with similar flexibility) or `histplot` (an axes-  
level function for histograms).

warnings.warn(msg, FutureWarning)

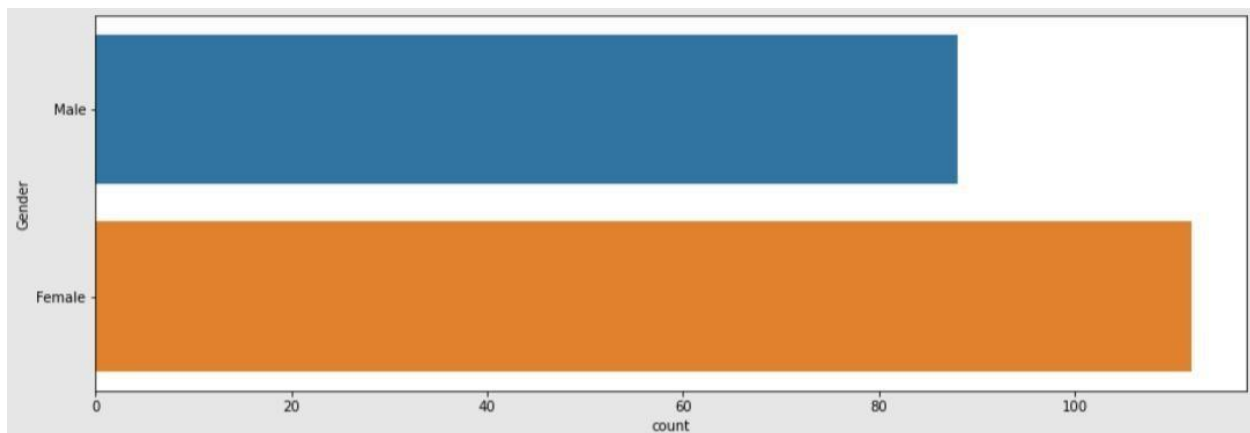
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557:  
FutureWarning: `distplot` is a deprecated function and will be removed  
in a future version. Please adapt your code to use either `displot` (a  
figure-level function with similar flexibility) or `histplot` (an axes-  
level function for histograms). warnings.warn(msg, FutureWarning)

/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557:  
FutureWarning: `distplot` is a deprecated function and will be removed  
in a future version. Please adapt your code to use either `displot` (a  
figure-level function with similar flexibility) or `histplot` (an axes-  
level function for histograms). warnings.warn(msg, FutureWarning)



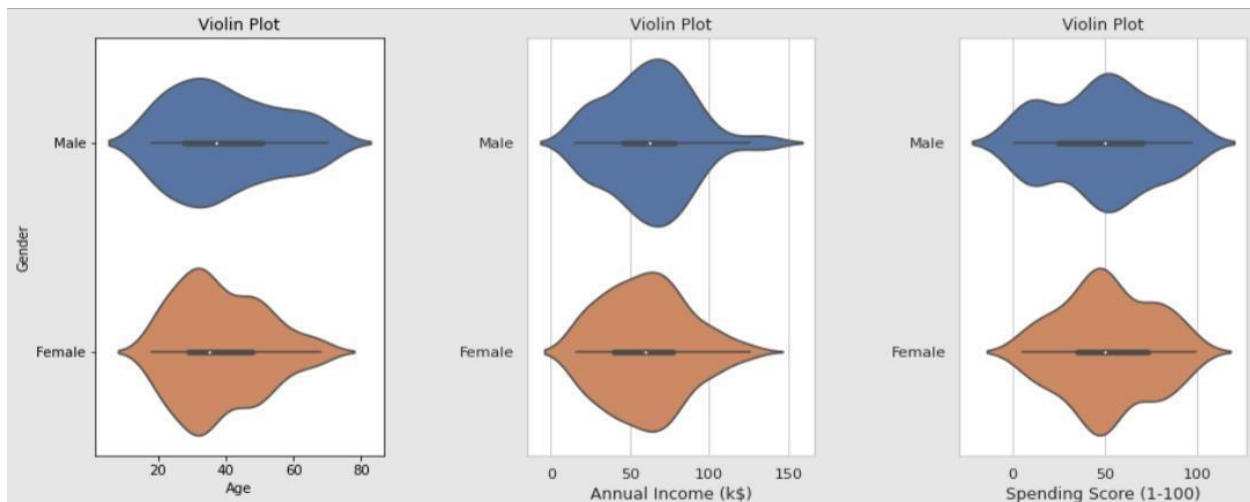
In [8]:

```
plt.figure(figsize=(15,5))  
sns.countplot(y='Gender',data=df) plt.show()
```



In [9]:

```
plt.figure(1,figsize=(15,6)) n = 0 for cols in ['Age', 'Annual
Income (k$)', 'Spending Score (1-100)']: n +=1
    plt.subplot(1,3,n)
sns.set(style="whitegrid")
    plt.subplots_adjust(hspace=0.5,wspace=0.5)
sns.violinplot(x = cols,y = 'Gender',data=df)
plt.ylabel('Gender' if n== 1 else '')
plt.title('Violin Plot') plt.show()
```



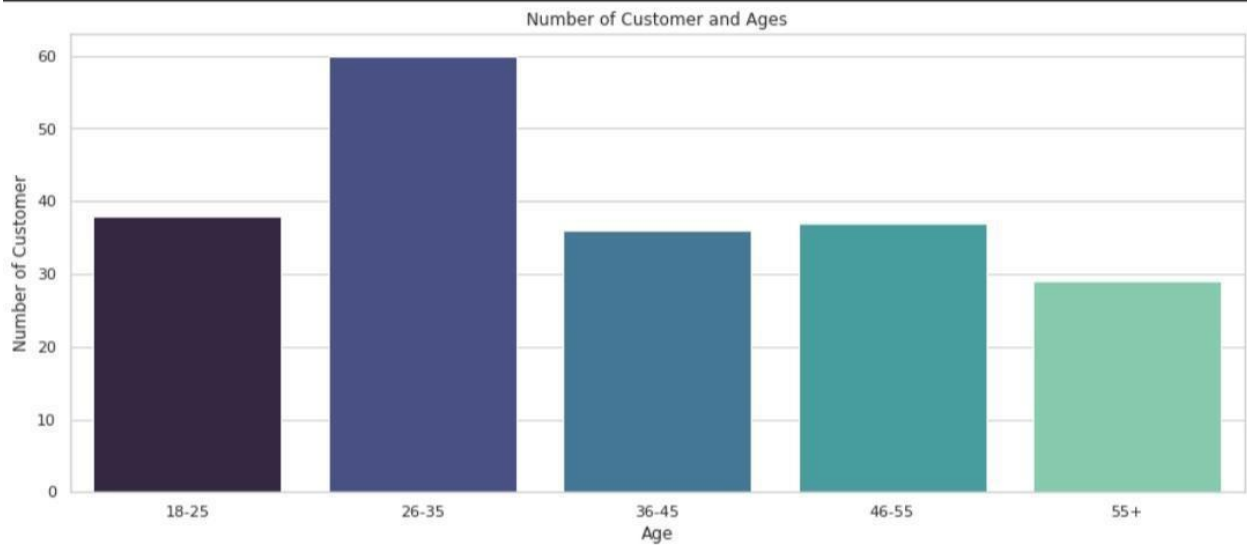
In [10]:

```
age_18_25 = df.Age[(df.Age >=18) & (df.Age <= 25)]
age_26_35 = df.Age[(df.Age >=26) & (df.Age <= 35)]
age_36_45 = df.Age[(df.Age >=36) & (df.Age <= 45)]
age_46_55 = df.Age[(df.Age >=46) & (df.Age <= 55)]
age_55_above = df.Age[(df.Age >= 56)]

age_x = ["18-25", "26-35", "36-45", "46-55", "55+"]
age_y =
[ len(age_18_25.values), len(age_26_35.values), len(age_36_45), len(ag
e_46_55), len(age_55_above)]

plt.figure(figsize = (15,6))
sns.barplot(x=age_x, y=age_y,palette = "mako")
plt.title("Number of Customer and Ages")
plt.xlabel("Age")
plt.ylabel("Number of Customer") plt.show()
```



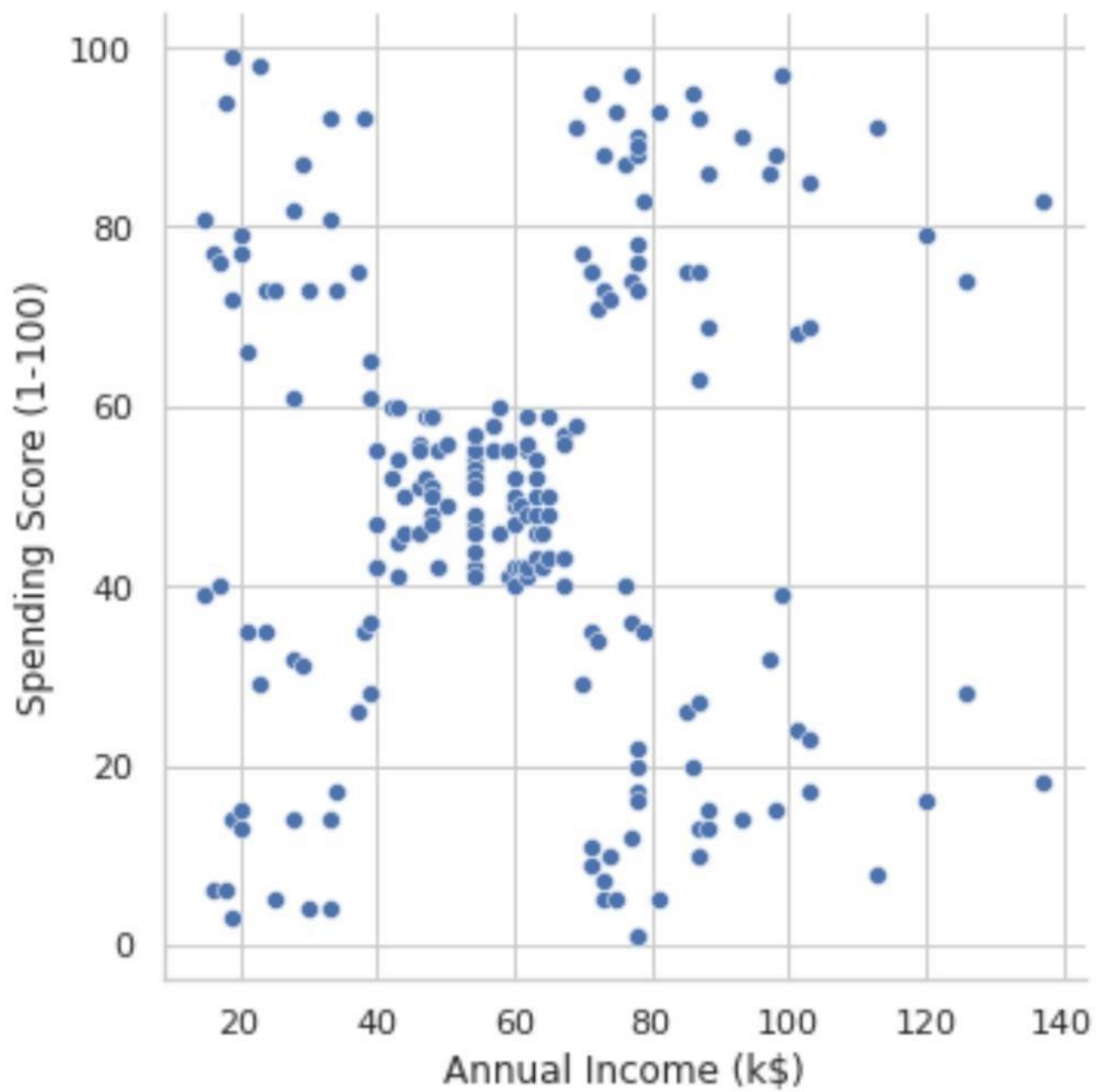


In [11]:

```
sns.relplot(x="Annual Income (k$)", y = "Spending Score (1-100)", data=df)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x7fcef0536fd0>
```



In [12]:

```

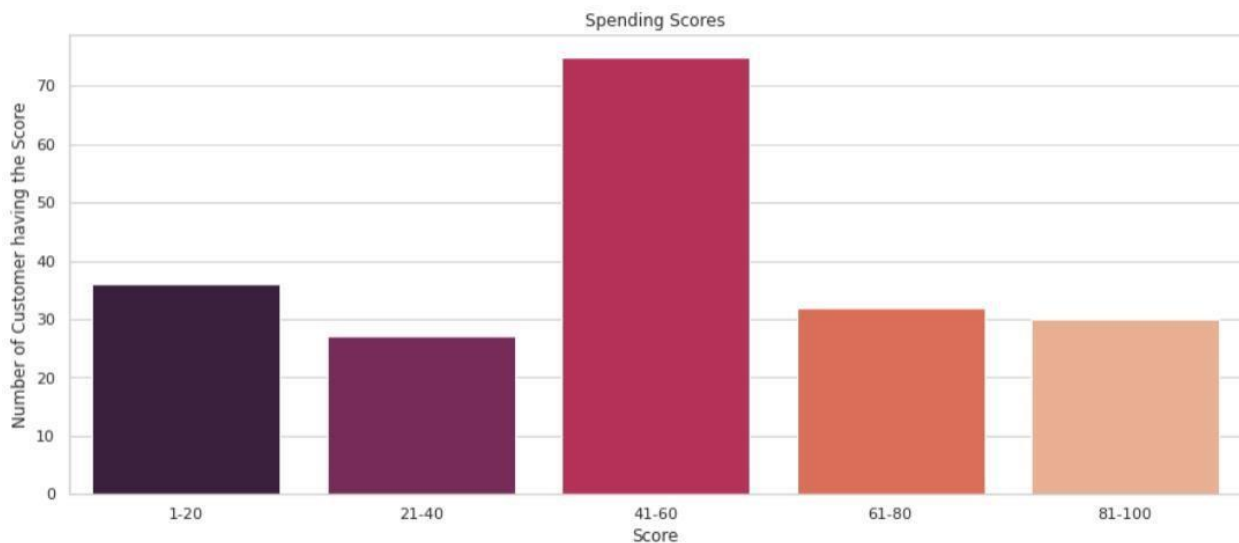
ss_1_20 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 1)
    & (df["Spending Score (1-100)"] <= 20)]
ss_21_40 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 21)
    & (df["Spending Score (1-100)"] <= 40)]
ss_41_60 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 41)
    & (df["Spending Score (1-100)"] <= 60)]
ss_61_80 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 61)
    & (df["Spending Score (1-100)"] <= 80)]

ss_81_100 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 81)
    & (df["Spending Score (1-100)"] <= 100)]

ssx= ["1-20", "21-40", "41-60", "61-80", "81-100"]
ssy=[len(ss_1_20.values), len(ss_21_40.values), len(ss_41_60.values), len(ss_61_80.values), len(ss_81_100.values)]

plt.figure(figsize=(15,6))
sns.barplot(x=ssx,y=ssy, palette="rocket")
plt.title("Spending Scores") plt.xlabel("Score")
plt.ylabel("Number of Customer having the Score")
plt.show()

```



In[13]:

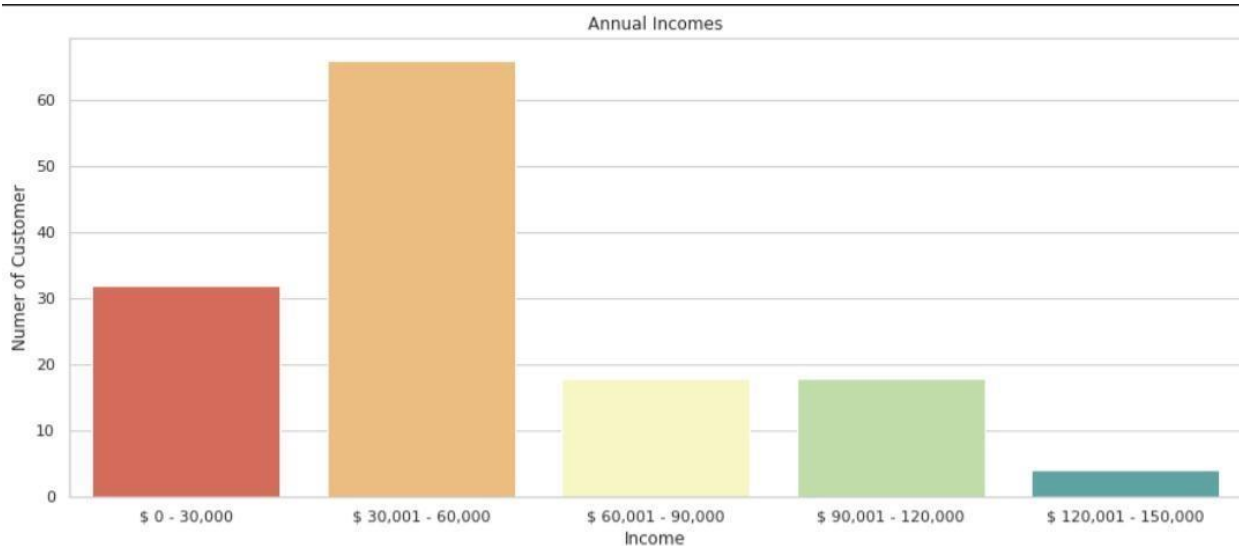
```

ai_0_30 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 0) &
(df[" Annual Income (k$)"] <= 30)]
ai_31_60= df["Annual Income (k$)"][(df["Annual Income (k$)"] >=31)&
(df["A nnual Income (k$)"] <=60)]
ai_61_90= df["Annual Income (k$)"][(df["Annual Income (k$)"] >=61)&
(df["A nnual Income (k$)"] <=90)]
ai_61_90=df["Annual Income (k$)"][(df["Annual Income (k$)"] >=91)& (df["An
nual Income (k$)"] <=120)]
ai_121_150 = df["Annual Income (k$)"][(df["Annual Income (k$)"]>=121) & (d
f["Annual Income (k$)"] <=150)]

aix = ["$ 0 - 30,000", "$ 30,001 - 60,000", "$ 60,001 - 90,000", "$ 90,001 -
120,000", "$ 120,001 - 150,000"]
aiy =
[ len(ai_0_30.values), len(ai_31_60.values), len(ai_61_90.values), len(a
i_61_90.values), len(ai_121_150.values)]

plt.figure(figsize=(15,6))
sns.barplot(x=aix,y=aiy,palette="Spectral")
plt.title("Annual Incomes") plt.xlabel("Income")
plt.ylabel("Numer of Customer")
plt.show()

```



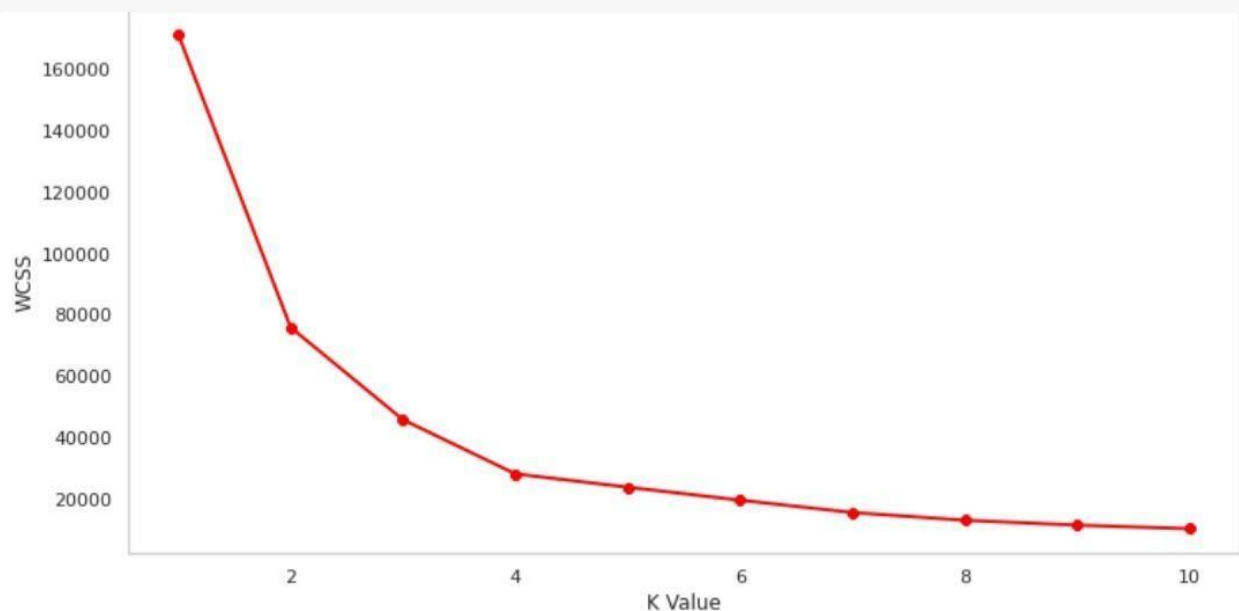
In [14]:

```

X1 = df.loc[:,["Age","Spending Score (1-100)"]].values

from sklearn.cluster import KMeans wcss=[] for k in
range(1,11):      kmeans = KMeans(n_clusters = k, init =
"k-means++")      kmeans.fit(X1)
      wcss.append(kmeans.inertia_)
plt.figure(figsize =( 12,6)) plt.grid()
plt.plot(range(1,11),wcss,linewidth=2,color="red",marker="8")
plt.xlabel("K Value") plt.ylabel("WCSS") plt.show()

```



```

In [15]:
kmeans = KMeans(n_clusters=4)

label = kmeans.fit_predict(X1)
print(label)

```

```

[1 2 0 2 1 2 0 2 0 2 0 2 0 2 0 2 1 1 0 2 1 2 0 2 0 2 0 1 0 2 0 2 0 2 0 2
0
2 0 2 3 2 3 1 0 1 3 1 1 1 3 1 1 3 3 3 3 1 3 3 1 3 3 3 1 3 3 1 1 3 3 3 3
3 1 3 1 1 3 3 1 3 3 1 3 3 1 1 3 3 1 3 1 1 3 1 3 1 1 3 3 1 3 1 3 3 3 3 3
1 1 1 1 1 3 3 3 3 1 1 1 2 1 2 3 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 1 2 0 2 3
2
0 2 0 2 0 2 0 2 0 2 0 2 3 2 0 2 0 2 0 2 0 1 0 2 0 2 0 2 0 2 0 2 0 2 0 2
1 2 0 2 0 2 0 2 0 2 0 2 0 2]

```

In [16]:

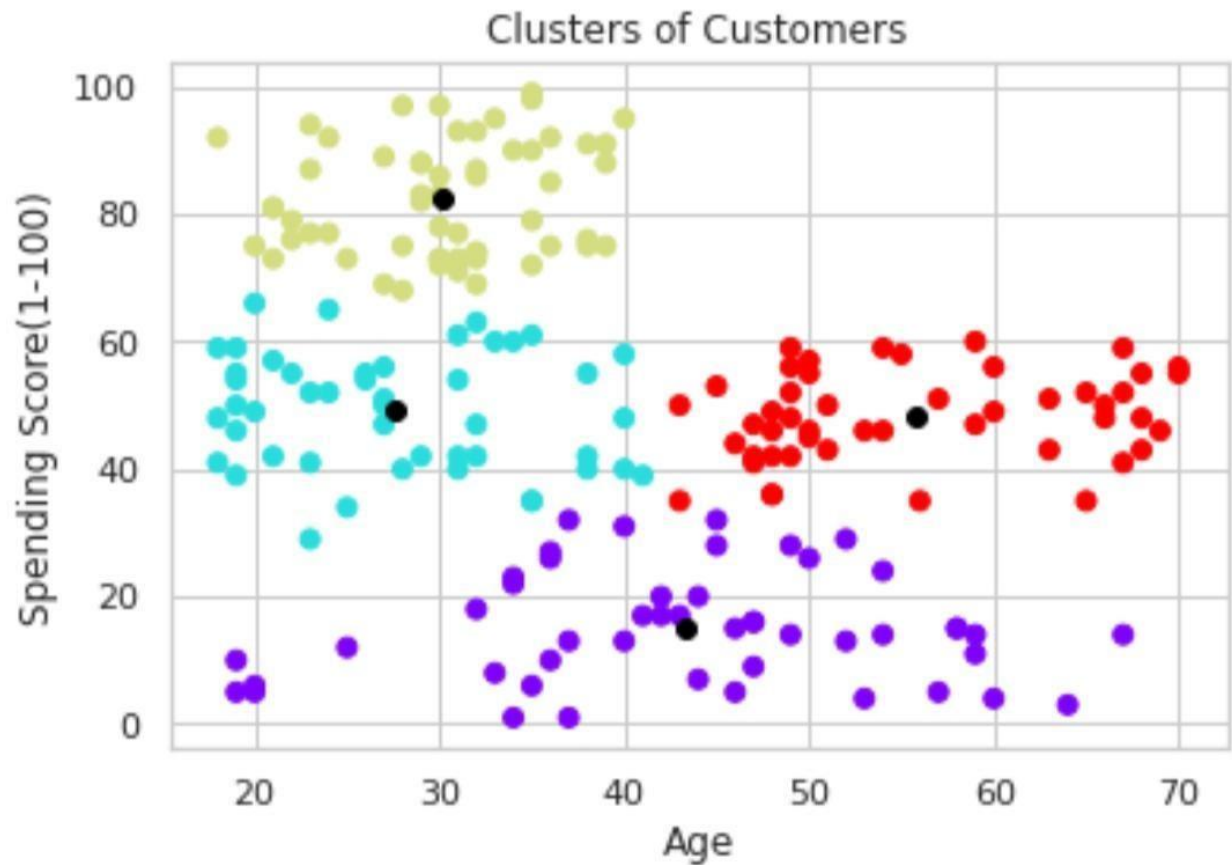
```
[[43.29166667 15.02083333]
 [27.61702128 49.14893617]
 [30.1754386  82.35087719]
 [55.70833333 48.22916667]]
```

In [17]:

```
plt.scatter(X1[:,0],X1[:,1],c=kmeans.labels_,cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[ :,0],kmeans.cluster_centers_[ :,1],color='black')
plt.title('Clusters of Customers')
plt.xlabel('Age')
plt.ylabel('Spending Score(1-100)') plt.show
```

Out[17]:

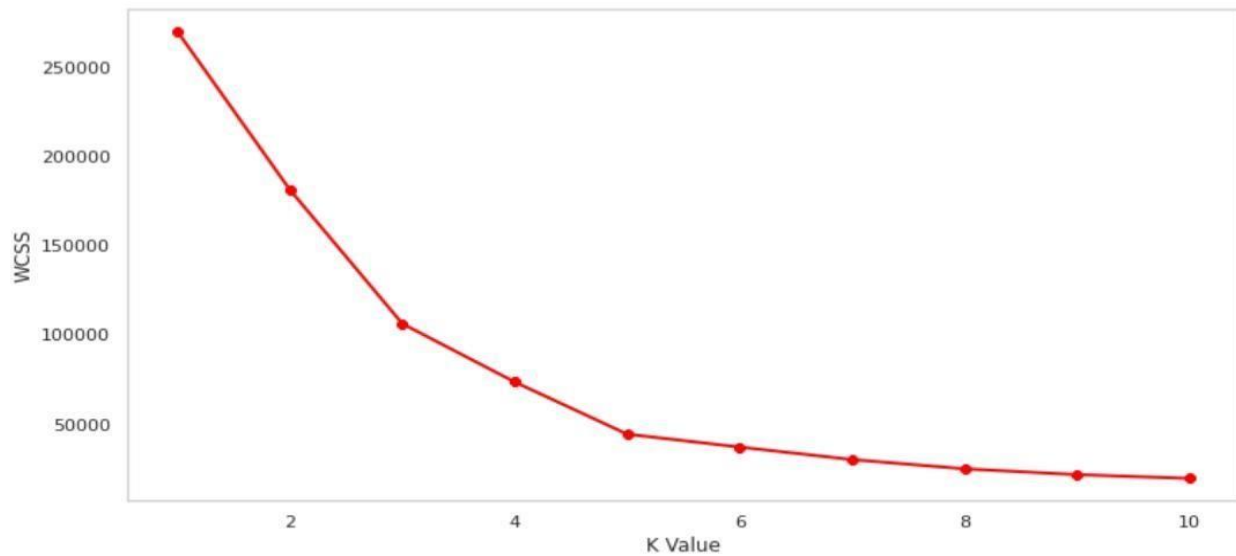
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [18]:

```
X2 = df.loc[:,["Annual Income (k$)","Spending Score (1-100)"]].values

from sklearn.cluster import KMeans wcss=[] for k in
range(1,11): kmeans = KMeans(n_clusters = k, init =
"k-means++") kmeans.fit(X2)
wcss.append(kmeans.inertia_)
plt.figure(figsize =( 12,6)) plt.grid()
plt.plot(range(1,11),wcss,linewidth=2,color="red",marker="8")
plt.xlabel("K Value") plt.ylabel("WCSS") plt.show()
```



In [19]:

```
kmeans = KMeans(n_clusters=5)
```

```
label = kmeans.fit_predict(X2)
```

```
print(label)
```

[illegible]

In [20]:

```
[ [25.72727273 79.36363636]
  [88.2         17.11428571]
  [55.2962963  49.51851852]
  [86.53846154 82.12820513]
  [26.30434783 20.91304348] ]
```

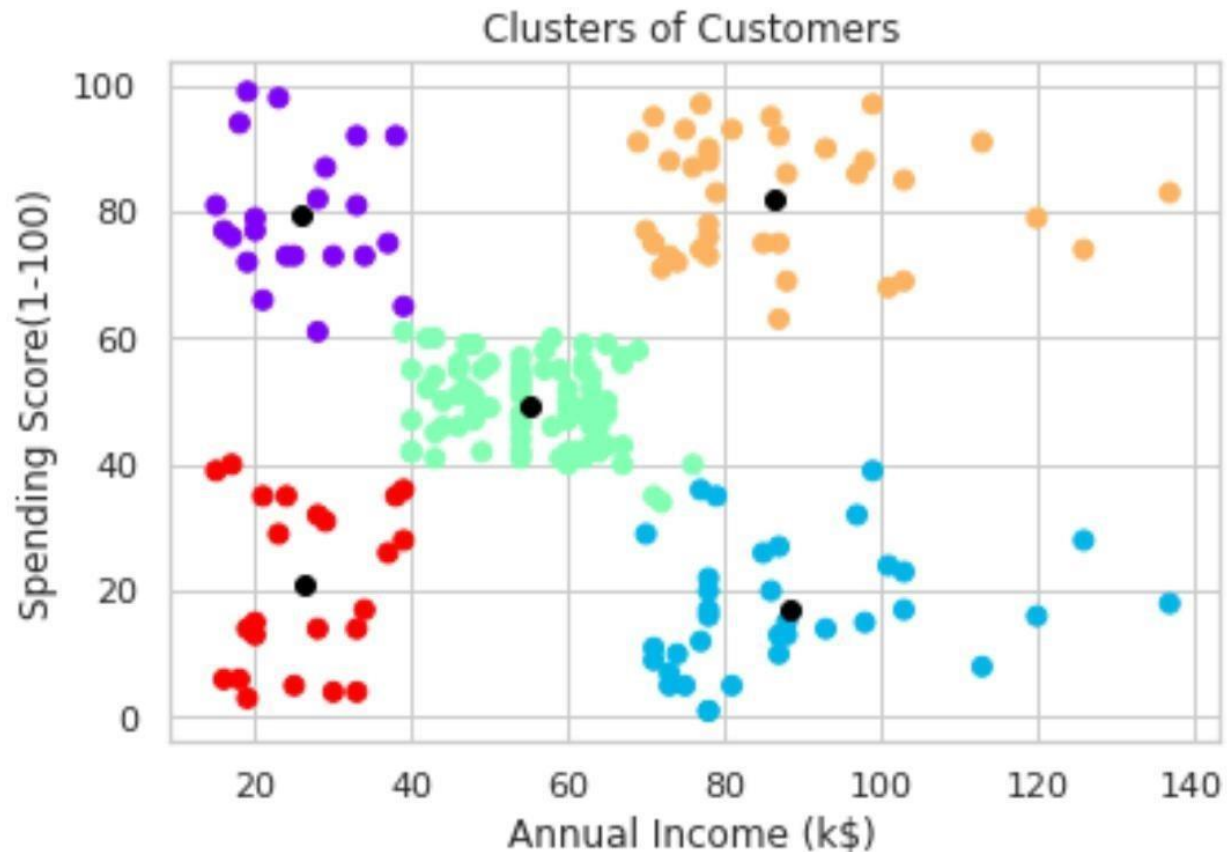
In [21]:

```
plt.scatter(X2[:,0],X1[:,1],c=kmeans.labels_,cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[ :,0],kmeans.cluster_centers_[ :,1],color='black')
plt.title('Clusters of Customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score(1-100)') plt.show
```



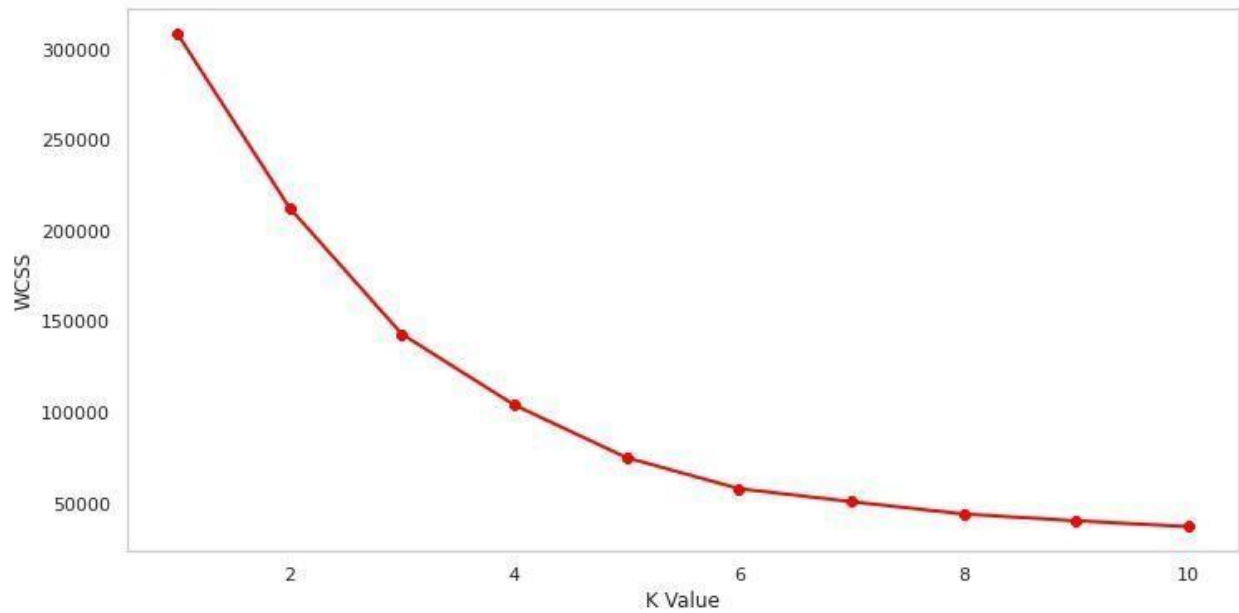
Out[21]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [22]:

```
X3 = df.iloc[:,1:]
wcss=[] for k in range(1,11):      kmeans =
KMeans(n_clusters = k, init = "k-means++")
kmeans.fit(X3)
wcss.append(kmeans.inertia_)
plt.figure(figsize =( 12,6)) plt.grid()
plt.plot(range(1,11),wcss,linewidth=2,color="red",marker="8")
plt.xlabel("K Value") plt.ylabel("WCSS") plt.show()
```



In [23]:

```
kmeans = KMeans(n_clusters=5)
```

```
label = kmeans.fit_predict(X3)
```

```
print(label)
```

[illegible]

In [24]:

```
[ [43.08860759 55.29113924 49.56962025]
  [25.52173913 26.30434783 78.56521739]
  [40.66666667 87.75          17.58333333]
  [45.2173913  26.30434783 20.91304348]
  [32.69230769 86.53846154 82.12820513] ]
```

In [25]:

```

cluster = kmeans.fit_predict(X3) df["label"]
= cluster

from mpl_toolkits.mplot3d import Axes3D

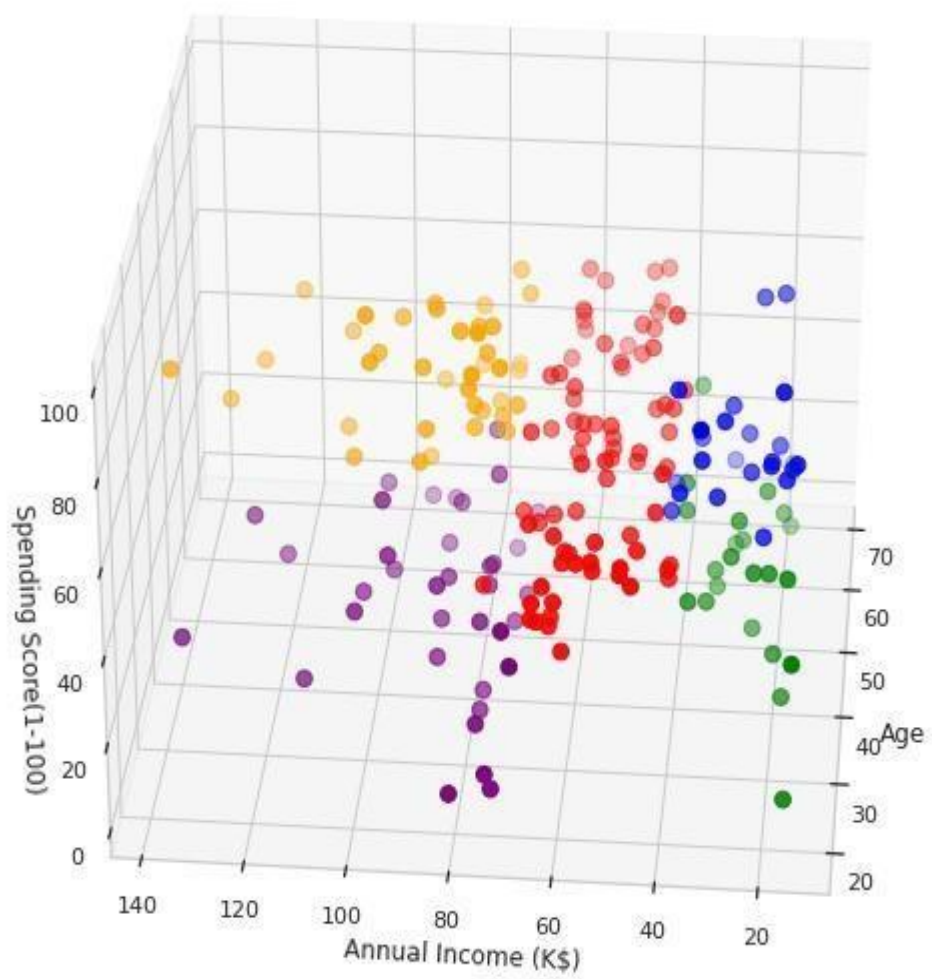
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111,projection = '3d')

ax.scatter(df.Age[df.label == 0 ],df["Annual Income (k$)"][df.label == 0],
df["Spending Score (1-100)"][df.label == 0], c = 'blue',s=60)
ax.scatter(df.Age[df.label == 1 ],df["Annual Income (k$)"][df.label == 1],
df["Spending Score (1-100)"][df.label == 1], c = 'red',s=60)
ax.scatter(df.Age[df.label == 2 ],df["Annual Income (k$)"][df.label == 2],
df["Spending Score (1-100)"][df.label == 2], c = 'green',s=60)
ax.scatter(df.Age[df.label == 3 ],df["Annual Income (k$)"][df.label ==
3], df["Spending Score (1-100)"][df.label == 3], c = 'orange',s=60)
ax.scatter(df.Age[df.label == 4 ],df["Annual Income (k$)"][df.label ==
4], df["Spending Score (1-100)"][df.label == 4], c = 'purple',s=60)

ax.view_init(30,185)

plt.xlabel("Age")
plt.ylabel("Annual Income (K$)")
ax.set_zlabel('Spending Score(1-100)')
plt.show()

```



## ADVANTAGES:

- **Targeted Marketing:** Customer segmentation helps in identifying specific subsets of customers with similar characteristics, enabling businesses to tailor their marketing campaigns and messages to each segment. This increases the effectiveness of marketing efforts and drives better results.
- **Improved Customer Satisfaction:** By understanding the unique needs and preferences of different customer segments, businesses can provide personalized experiences and offerings. This leads to higher customer satisfaction and loyalty, as customers feel understood and valued by the company.
- **Cost Efficiency:** Customer segmentation allows businesses to allocate their marketing resources more efficiently. Instead of indiscriminately targeting a large audience, companies can focus their efforts on the most promising customer segments. This saves costs on marketing campaigns and improves overall ROI.
- **Product and Service Development:** Insights gained from customer segmentation can inform product and service development strategies. Businesses can identify gaps in the market and develop offerings that cater specifically to the needs of different customer segments, leading to innovation and competitive advantage.
- **Customer Retention and Acquisition:** Customer segmentation helps in identifying high-value segments and understanding their behavior and preferences. This knowledge can be used to implement retention strategies for existing customers and acquisition strategies for new customers who match the characteristics of the most profitable segments.
- **Data-Driven Decision Making:** Customer segmentation relies on data analysis and statistical techniques to identify patterns and trends. This enables businesses to make informed decisions based on data-driven insights rather than relying solely on intuition or assumptions.

- **Market Differentiation:** By targeting specific customer segments, businesses can differentiate themselves from competitors. They can position their products or services as uniquely tailored to meet the needs of a particular segment, setting themselves apart in the market and attracting loyal customers.

Overall, customer segmentation using data science enables businesses to better understand their customers, deliver personalized experiences, optimize marketing efforts, and ultimately drive business success.

## DISADVANTAGES:

- **Privacy Concerns:** Data science relies on collecting and analyzing customer data, which can raise privacy concerns. Customers may feel uncomfortable knowing that their personal information is being used for segmentation purposes.
- **Data Accuracy and Quality:** The effectiveness of customer segmentation relies on the accuracy and quality of the data being used. If the data is incomplete, outdated, or inaccurate, it can lead to flawed segmentation and ineffective marketing strategies.
- **Bias and Stereotyping:** Using data science for customer segmentation may unintentionally reinforce biases or stereotypes. If the data used for segmentation is biased, it can lead to discriminatory practices or unfair targeting of certain customer groups.
- **Overreliance on Data:** Relying solely on data for segmentation may overlook other important factors that can contribute to customer behavior. Human intuition and qualitative insights may be disregarded in favor of data-driven decisions, potentially missing out on valuable insights.
- **Complexity and Implementation Challenges:** Implementing data science for customer segmentation requires specialized skills, technology, and resources. It can be a complex process that requires a deep understanding of data analysis techniques and algorithms, making it challenging for some businesses to adopt and implement effectively.

- **Limited Customer Understanding:** While data science provides quantitative insights into customer behavior, it may lack a deeper understanding of customer motivations, emotions, and preferences. It may miss the "why" behind customer actions, limiting the ability to create personalized and meaningful customer experiences.

It is important for businesses to carefully consider these disadvantages and address them transparently and ethically when using data science for customer segmentation.

## **BENEFITS :**

Customer segmentation using data science offers several key benefits, including:

- **Improved Personalization:** By identifying distinct customer groups, businesses can tailor their offerings and marketing strategies to better meet the specific needs and preferences of each segment.
- **Enhanced Customer Retention:** Understanding customer behaviors and preferences enables businesses to implement targeted retention strategies, reducing churn and fostering long-term customer relationships.
- **Optimized Marketing Efforts:** Data-driven segmentation allows businesses to allocate resources effectively, focusing marketing efforts on the most promising customer segments, which can lead to higher conversion rates and improved ROI.
- **Product Development:** Insights gained through customer segmentation can guide product development, ensuring that new offerings align with the needs and preferences of specific customer groups.

- **Increased Customer Satisfaction:** By providing personalized experiences and relevant products or services, businesses can enhance overall customer satisfaction, leading to higher customer loyalty and positive brand perception.
- **Cost Efficiency:** Targeted marketing and product development based on segmentation data can lead to cost savings by avoiding unnecessary expenditures on ineffective strategies or irrelevant product features.

Overall, customer segmentation using data science enables businesses to make data-backed decisions, leading to more effective and efficient customer engagement, improved customer satisfaction, and ultimately, increased profitability.

## REFERENCE:

Customer segmentation using data science is a widely studied and applied concept in the field of marketing and business. Some notable references include:

- **"Customer Segmentation and Clustering Using SAS Enterprise Miner"** by Randall S. Collica: This book provides a comprehensive guide to customer segmentation techniques using SAS Enterprise Miner, offering practical insights into the application of data science in customer segmentation.
- **"Data-Driven Marketing: The 15 Metrics Everyone in Marketing Should Know"** by Mark Jeffery: This book discusses various data-driven marketing strategies, including customer segmentation, and emphasizes the importance of using data science to optimize marketing efforts.
- **"Customer Segmentation and Clustering Using Python"** by Sandro Saitta: This resource focuses on applying Python for customer segmentation and clustering, providing practical examples and case studies to demonstrate the use of data science techniques in customer segmentation.



These references offer valuable insights into the theory and application of customer segmentation using data science, providing guidance and best practices for businesses looking to leverage data-driven strategies to understand and target their customer base more effectively.

## CONCLUSION:

Customer segmentation using data science follows several key steps that culminate in effective segmentation strategies. Here's a concise step-by-step conclusion:

- **Data Collection and Preprocessing:** Gather relevant customer data from various sources and clean, preprocess, and organize it for analysis.
- **Exploratory Data Analysis (EDA):** Conduct a thorough examination of the data to identify patterns, trends, and potential segments within the customer base.
- **Feature Selection and Engineering:** Choose meaningful features that can effectively differentiate between customer segments and create new features that enhance the understanding of customer behavior.
- **Model Selection and Application:** Choose appropriate data science models such as clustering algorithms (e.g., k-means, hierarchical clustering) or classification algorithms (e.g., decision trees, random forests) to segment the customer base based on identified features.
- **Evaluation of Segmentation Results:** Evaluate the effectiveness of the segmentation by assessing how well the segments align with the predefined business objectives and whether they provide actionable insights for targeted marketing and product development.

- **Implementation and Monitoring:** Implement the segmentation strategy in marketing campaigns and product development initiatives. Continuously monitor and refine the segmentation based on new data and feedback to ensure its ongoing relevance and effectiveness.

By following these steps, businesses can effectively leverage data science for customer segmentation, enabling them to better understand their customer base, tailor their offerings, and optimize marketing strategies to drive business growth and enhance customer satisfaction.