Name: Prathamesh Mahesh BHangale

Registration Number: 21BCI0285

Email ID: prathamesh.mahesh2021@vitstudent.ac.in

Campus: VIT Vellore

Branch: Computer Science and Engineering

Phone Number: 9373419818

Assignment 5

Market Basket Magic: Extracting Insights for Retail Success

Task 1: Download the Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns
from scipy import stats
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/Mall_Customers.csv')
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

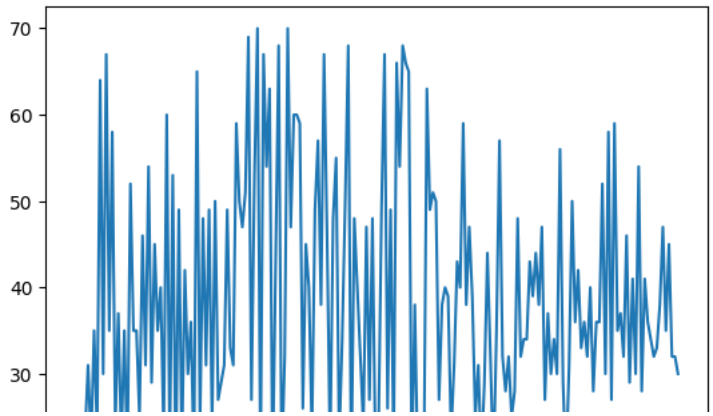
Task 2: Data Pre-Processing

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

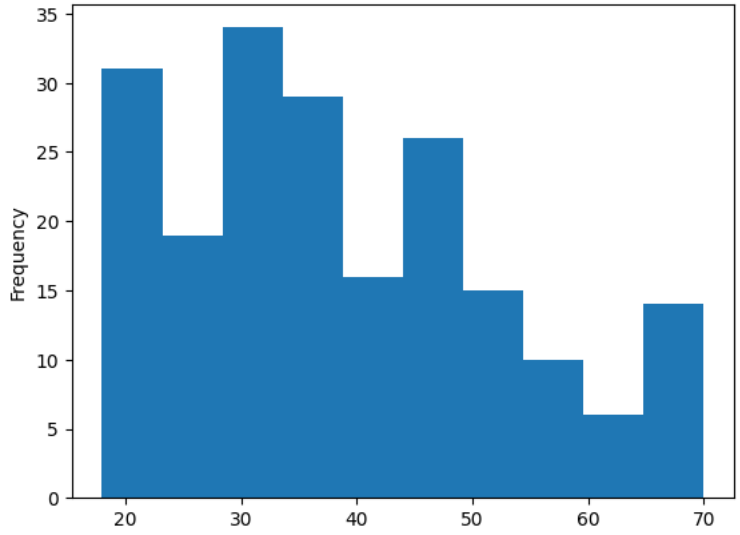
```
# We perform various visualizations :
# a) Univariate Analysis

# --------------> Univariate Analysis of Age <----------------------
df['Age'].plot()
```

```
<Axes: >
```



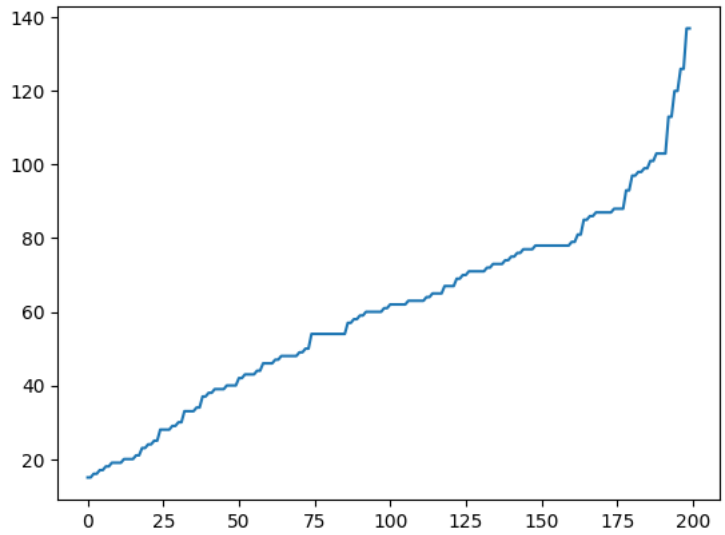

```
df['Age'].plot(kind = 'hist')
```

```
<Axes: ylabel='Frequency'>
```



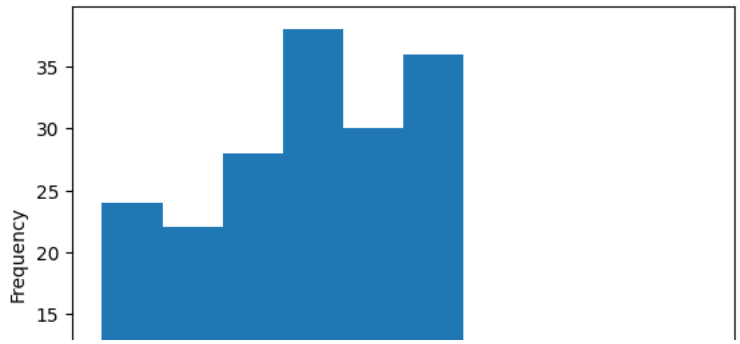

```
# --------------> Univariate Analysis of Annual Income (k$) <----------------------
df['Annual Income (k$)'].plot()
```

```
<Axes: >
```



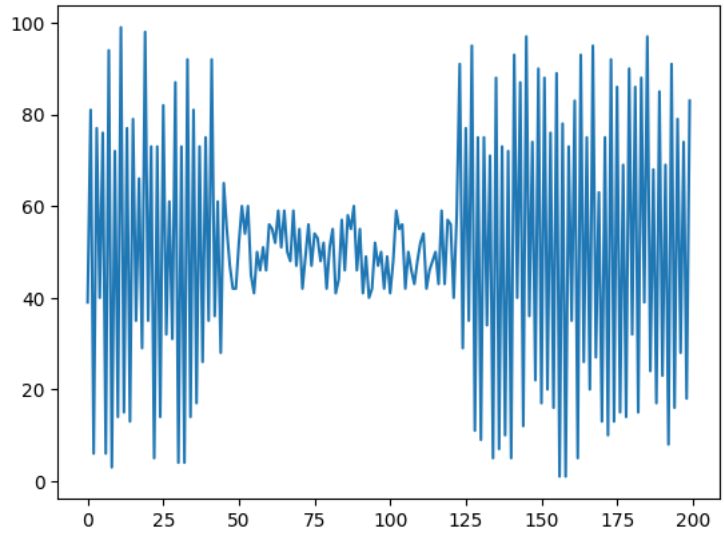

```
df['Annual Income (k$)'].plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



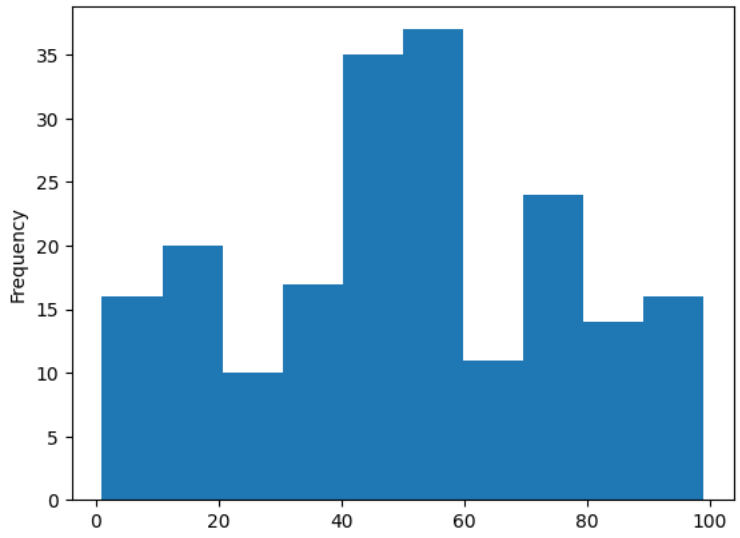

```
# --------------> Univariate Analysis of Spending Score (1-100)) <----------------------
df['Spending Score (1-100)'].plot()
```

```
<Axes: >
```



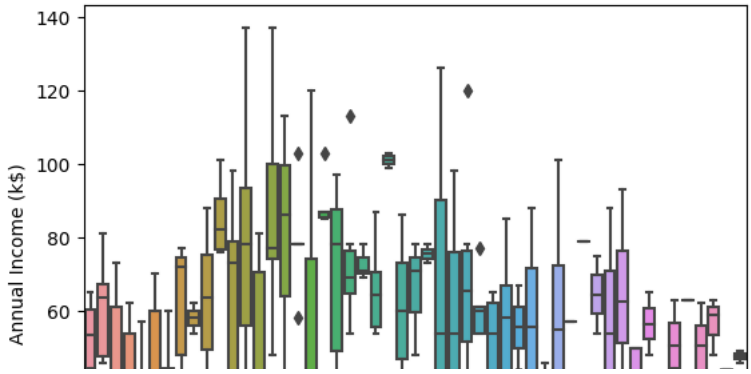

```
df['Spending Score (1-100)'].plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



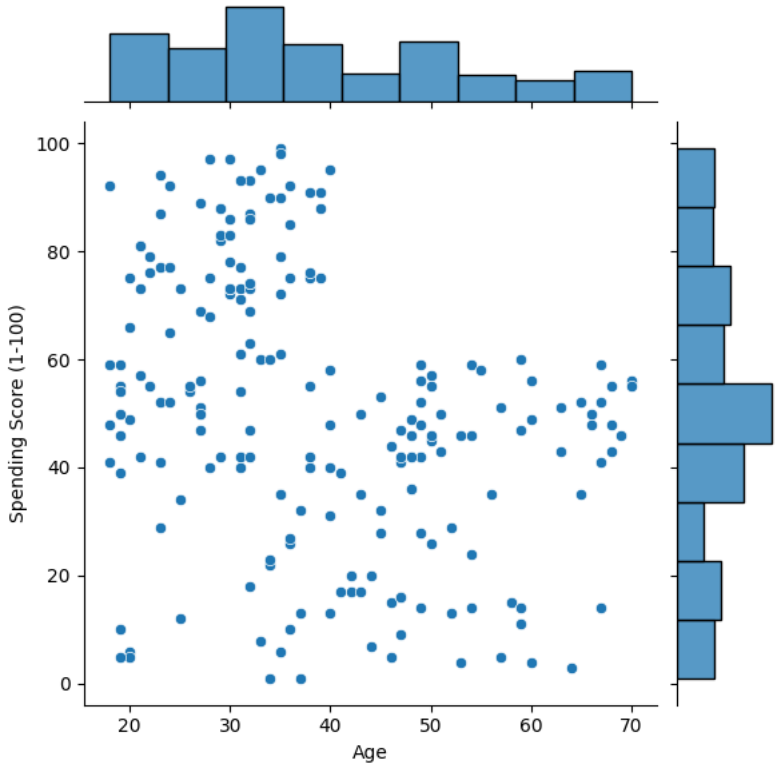

```
# b) Bi-Variate Analysis

# --> Bivariate Analysis of Age and Annual Income (k$) <-----
sns.boxplot(data=df, x='Age', y='Annual Income (k$)')
```

```
<Axes: xlabel='Age', ylabel='Annual Income (k$)'>
```



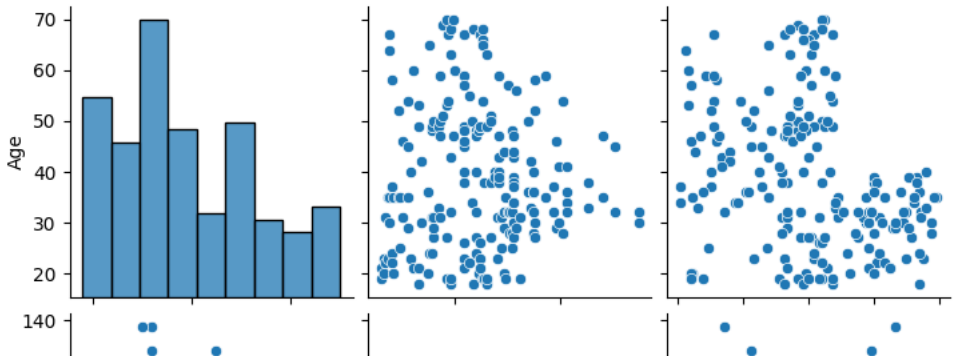

```
# --> Bivariate Analysis of Age and Spending Score (1-100) <-----
sns.jointplot(data=df, x='Age', y='Spending Score (1-100)')
```

```
<seaborn.axisgrid.JointGrid at 0x7ac6beca59f0>
```




```
# c) Multi-Variate Analysis

sns.pairplot(df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']])
```

```
<seaborn.axisgrid.PairGrid at 0x7ac6bfe45960>
```




```
sns.heatmap(df.corr())
```

```
<ipython-input-18-aa4f4450a243>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  sns.heatmap(df.corr())
<Axes: >
```


```
#Perform descriptive statistics on the dataset.
df.describe()
```

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

```
#Handle the missing values
#We handle the missing values by inserting the mean value of the respective columns in that missing vlaues

#Step 1: Identify which columns have missing values and how many in each column:
print(df.isnull().sum())
```

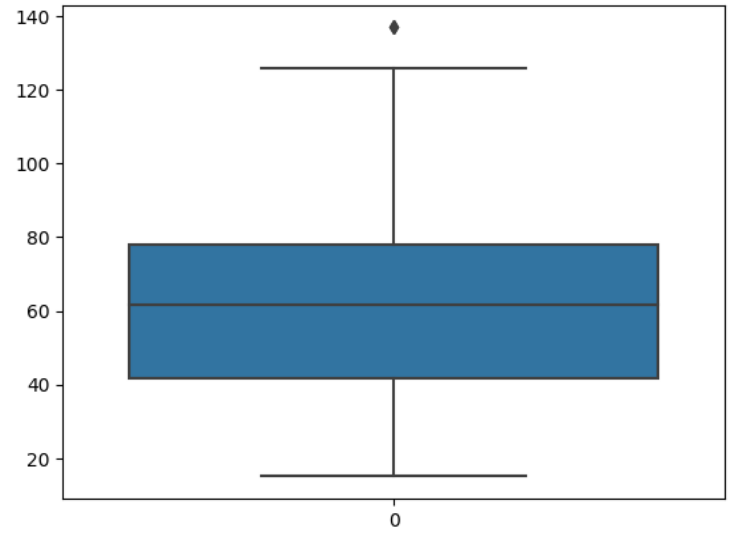
```
CustomerID              0
Gender                  0
Age                     0
Annual Income (k$)      0
```

```
Spending Score (1-100)    0
dtype: int64
```

As we can see there are no missing values in the given dataset

```
#Outlier Detection for Annual Income (k$)
import seaborn as sb
sb.boxplot(df['Annual Income (k$)'])
```

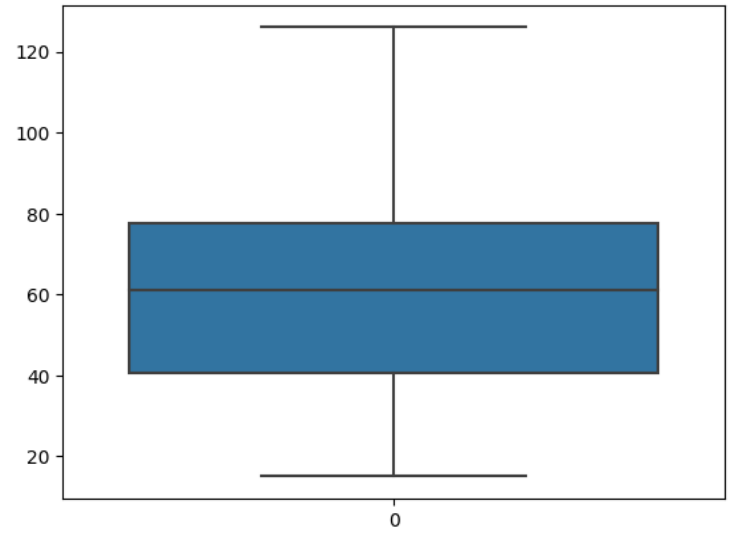
```
<Axes: >
```




```
# Outlier Removal
Q1 = df['Annual Income (k$)'].quantile(0.25)
Q3 = df['Annual Income (k$)'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR

upper_array = np.where(df['Annual Income (k$)']>=upper)[0]
lower_array = np.where(df['Annual Income (k$)']<=lower)[0]

# Removing the outliers
df.drop(index=upper_array, inplace=True)
df.drop(index=lower_array, inplace=True)
```

```
#Checking if the outliers have been removed
sb.boxplot(df['Annual Income (k$)'])
```

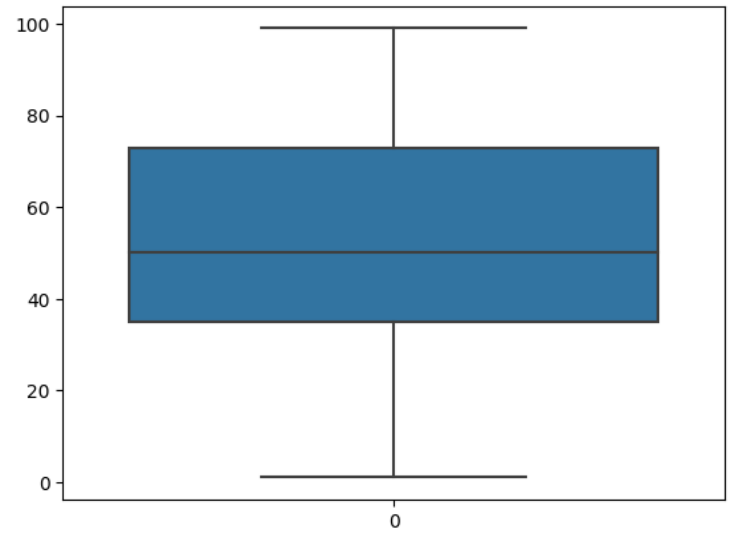
```
<Axes: >
```



The outliers have been successully removed

```
#Outlier Detection for Spending Score (1-100)
sb.boxplot(df['Spending Score (1-100)'])
```

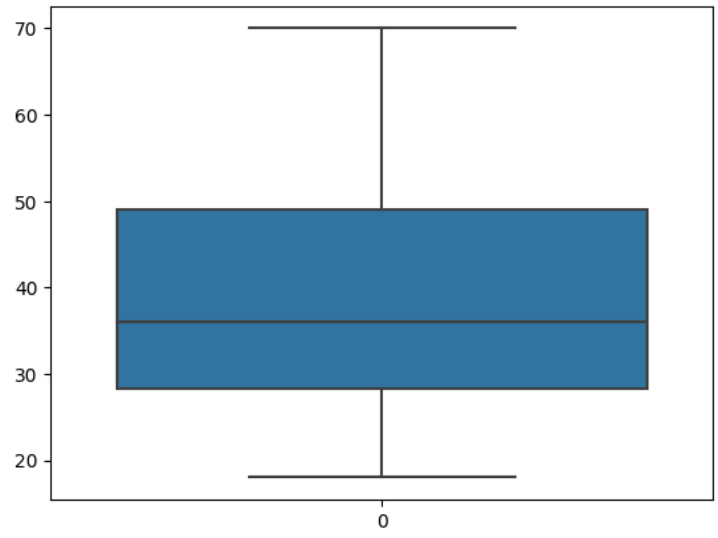
```
<Axes: >
```



As we can see there are no outliers

```
#Outlier Detection for Age
sb.boxplot(df['Age'])
```

```
<Axes: >
```



No outliers here too.

```
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Now we Encode the Gender column using Label Encoding

```
from sklearn.preprocessing import LabelEncoder
le =LabelEncoder()
df.Gender = le.fit_transform(df.Gender)
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40

Task 3: Machine Learning Model Building with Clustering Algorithm

```
# Split the data into dependent and independent variables
X =df.iloc[:,:-1] # Independent variables (features)
X.head()
```

	CustomerID	Gender	Age	Annual Income (k$)
0	1	1	19	15
1	2	1	21	15
2	3	0	20	16
3	4	0	23	16
4	5	0	31	17

```
#Check the shape of X
print("Shape of X (features):", X.shape)
```

```
Shape of X (features): (198, 4)
```

```
y =df['Spending Score (1-100)'] # Dependent variable (target)
y.head()
```

```
0    39
1    81
2     6
3    77
4    40
Name: Spending Score (1-100), dtype: int64
```

```
# Check the shape of y

print("Shape of y (target):", y.shape)
```

```
Shape of y (target): (198,)
```

```
#Split the data into training and testing
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=10)

# Check the shapes of training and testing data
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (148, 4)
Shape of X_test: (50, 4)
Shape of y_train: (148,)
Shape of y_test: (50,)
```

Scaling Our Data

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

```
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40

Implementing Clustering Algorithm

```
from sklearn.cluster import KMeans
km = KMeans(n_clusters = 3)
km
```

```
▾      KMeans
KMeans(n_clusters=3)
```

```
km.fit(X)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
▾      KMeans
KMeans(n_clusters=3)
```

```
predict = km.predict(X)
```

```
predict
```

```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 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,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
      dtype=int32)
```

To visualise this better we can add an extra column named "Cluster" to our data to see how the data has been divided.

```
df['Cluster'] = predict
df
```

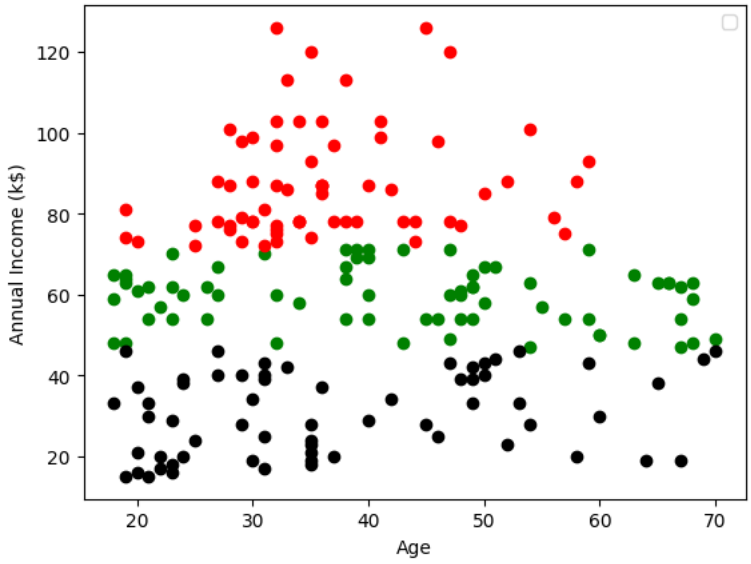
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	1	1	19	15	39	2
1	2	1	21	15	81	2
2	3	0	20	16	6	2
3	4	0	23	16	77	2
4	5	0	31	17	40	2
...	...	...	...	...	...	...
193	194	0	38	113	91	1
194	195	0	47	120	16	1
195	196	0	35	120	79	1
196	197	0	45	126	28	1
197	198	1	32	126	74	1

198 rows × 6 columns

```
#Let's see how it looks in scatter plot
# We Split it into 3 different data frames
```

```
df1 = df[df.Cluster == 0]
df2 = df[df.Cluster == 1]
df3 = df[df.Cluster == 2]


plt.scatter(df1['Age'],df1['Annual Income (k$)'],color='green')
plt.scatter(df2['Age'],df2['Annual Income (k$)'],color='red')
plt.scatter(df3['Age'],df3['Annual Income (k$)'],color='black')

plt.xlabel('Age')
plt.ylabel('Annual Income (k$)')
plt.legend()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note that artists whose label start with an underscore are
<matplotlib.legend.Legend at 0x7ac6b9c74760>
```



As we can see , our data has been divided into clusters successfully into 3 parts as shown in 3 different colours: Black, Green and Red.

```
# Test the model with random observation

km.predict([[1.1,2.3,4.2,4.1]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KMeans was fitte
  warnings.warn(
array([2], dtype=int32)
```

```
km.predict([[2.2,2.0,1.3,1.1]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KMeans was fitte
  warnings.warn(
array([2], dtype=int32)
```

Thus we have successfully implemented the Clustering Algorithm

Thank You 😀