CUSTOMER SEGMENTATION USING DATA SCIENCE
PHASE-3 SUBMISSION

```
# This Python 3 environment comes with many helpful analytics libraries inst
alled
# For example, here's several helpful packages to load
```

IN:

```
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 l
ist all files under the input directory

import os
for dirname, _, filenames in os.walk('/DATA SCIENCE/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 5GB to the current directory (/DATA SCIENCE/working/)
that gets preserved as output when you create a version using "Save & Run Al
l"
# You can also write temporary files to / DATA SCIENCE /temp/, but they won'
t be saved outside of the current session
```

In [2]:

```
# Supress Warnings
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

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

In [4]:

```
# Data display coustomization
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', -1)
```

In [5]:

```
# To perform Hierarchical clustering
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
```

```
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
```

In [6]:
```
# import all libraries and dependencies for machine learning
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.decomposition import IncrementalPCA
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
from math import isnan
```

Data Preparation

Data Loading

In [7]:
```
mall= pd.read_csv(r"/kaggle/input/customer-segmentation-tutorial-in-python/
Mall_Customers.csv")
mall.head()
```

Out[7]:

	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 [8]:
```
mall.shape
```

Out[8]:
```
(200, 5)
```

In [9]:

```
mall.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 [10]:

```
mall.describe()
```

Out[10]:

	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

Duplicate Check

Data Cleaning

Null Percentage: Columns

In [14]:
```
(mall.isnull().sum() * 100 / len(mall)).value_counts(ascending=False)
```

Out[14]:
```
0.0    5
dtype: int64
```

Null Count: Columns

In [15]:
```
mall.isnull().sum()
```

Out[15]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Null Percentage: Rows

In [16]:
```
(mall.isnull().sum(axis=1) * 100 / len(mall)).value_counts(ascending=False)
```

Out[16]:
```
0.0    200
dtype: int64
```

Null Count: Rows

In [17]:
```
mall.isnull().sum(axis=1).value_counts(ascending=False)
```

Out[17]:
```
0    200
dtype: int64
```

There are no missing / Null values either in columns or rows

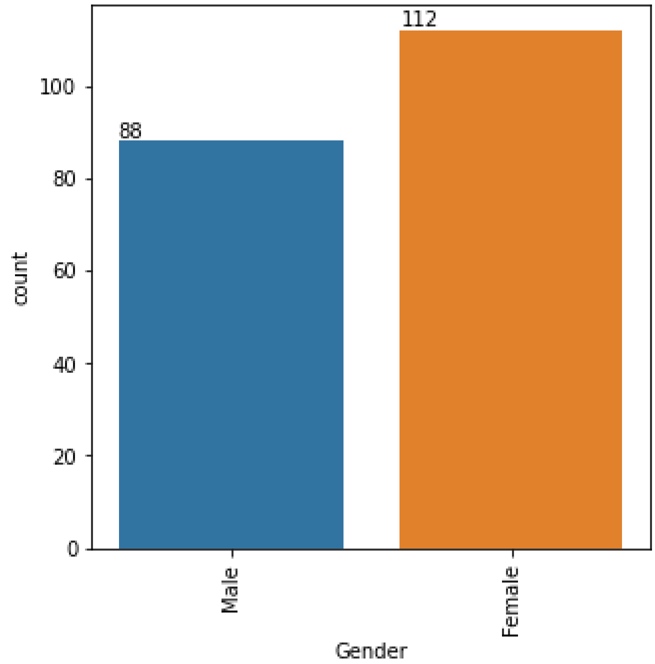
Exploratory Data Analytics

Univariate Analysis

Gender

In [18]:
```
plt.figure(figsize = (5,5))
gender = mall['Gender'].sort_values(ascending = False)
ax = sns.countplot(x='Gender', data= mall)
for p in ax.patches:
```

```
    ax.annotate(str(p.get_height()), (p.get_x() * 1.01 , p.get_height() * 1
.01))
plt.xticks(rotation=90)
plt.show()
```

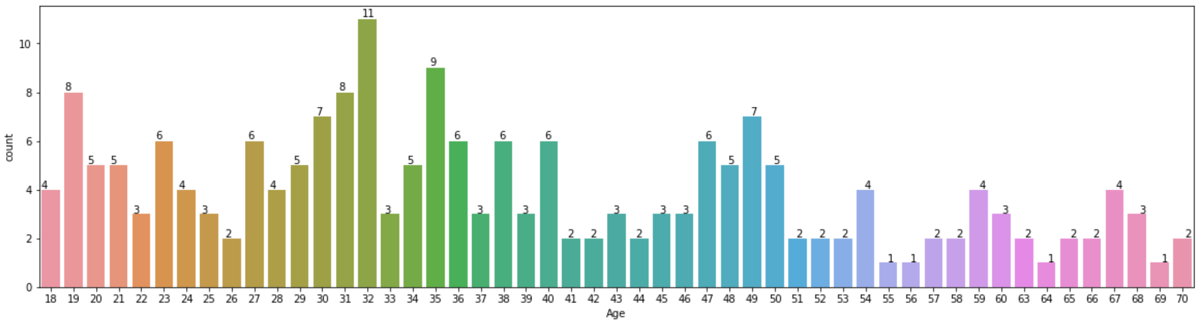


Data is not balanced, 6% more Females have participated than males

Age

In [19]:

```
plt.figure(figsize = (20,5))
gender = mall['Age'].sort_values(ascending = False)
ax = sns.countplot(x='Age', data= mall)
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.01 , p.get_height() * 1
.01))

plt.show()
```



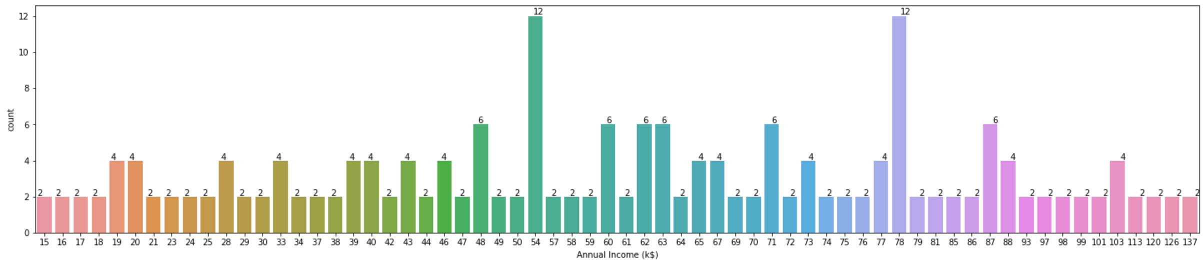
Audience are from Age 18 to 70

Annual Income (k$)

In [20]:

```
plt.figure(figsize = (25,5))
```

```
gender = mall['Annual Income (k$)'].sort_values(ascending = False)
ax = sns.countplot(x='Annual Income (k$)', data= mall)
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.01 , p.get_height() * 1
.01))

plt.show()
```

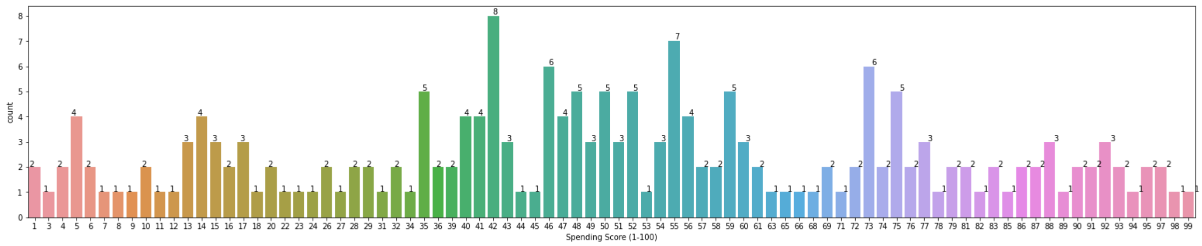


Audience are from Annual Income(k$) range between 15 to 137

Spending Score (1-100)

In [21]:

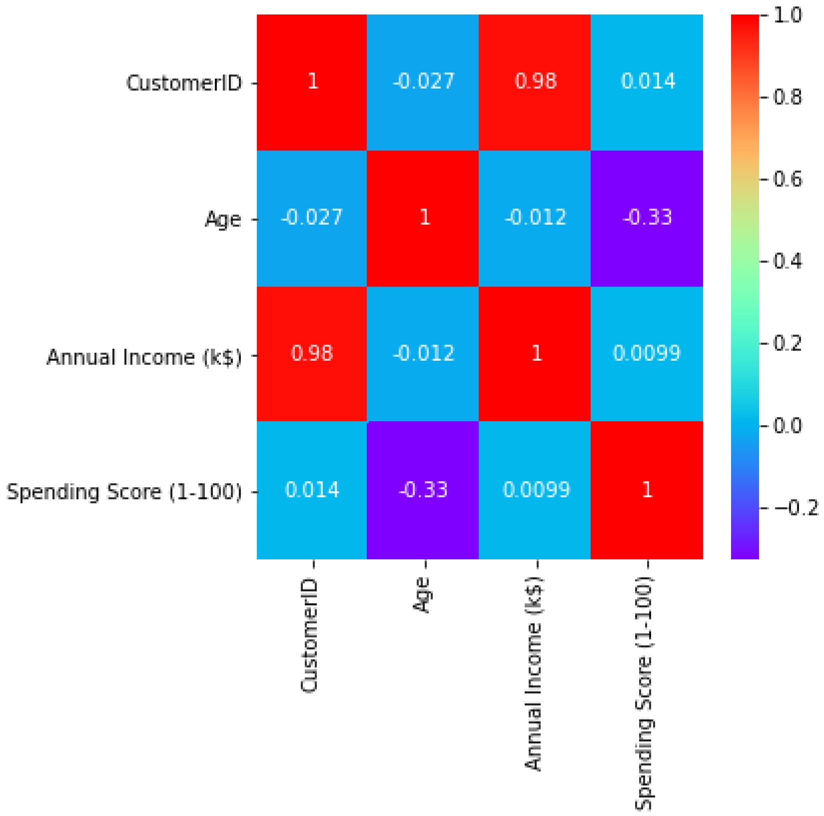
```
plt.figure(figsize = (27,5))
gender = mall['Spending Score (1-100)'].sort_values(ascending = False)
ax = sns.countplot(x='Spending Score (1-100)', data= mall)
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.01 , p.get_height() * 1
.01))

plt.show()
```



Audience are having Spending Score (1-100) between 1 to 99

In [22]:

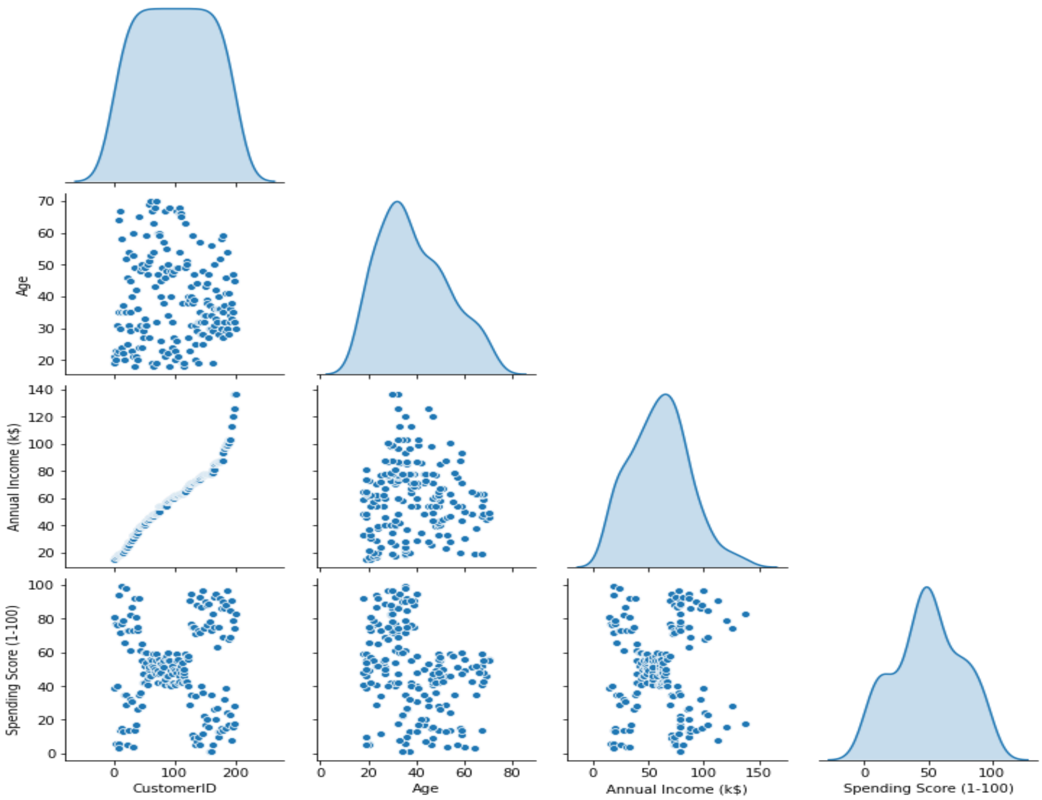
```
# Let's check the correlation coefficients to see which variables are highly
correlated

plt.figure(figsize = (5,5))
sns.heatmap(mall.corr(), annot = True, cmap="rainbow")
plt.savefig('Correlation')
plt.show()
```



- Age and Spending Score (1-100) are moderately correlated with correlation of -0.33

In [23]:

```
sns.pairplot(mall,corner=True,diag_kind="kde")
plt.show()
```



Outlier Analysis

In [24]:

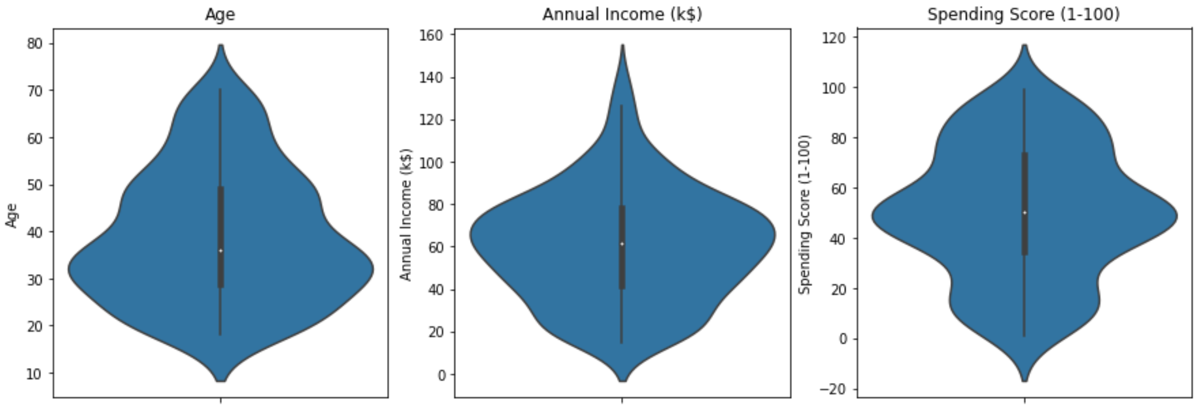
```
# Data before Outlier Treatment
mall.describe()
```

Out[24]:

	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 [25]:

```
f, axes = plt.subplots(1,3, figsize=(15,5))
s=sns.violinplot(y=mall.Age,ax=axes[0])
axes[0].set_title('Age')
s=sns.violinplot(y=mall['Annual Income (k$)'],ax=axes[1])
axes[1].set_title('Annual Income (k$)')
s=sns.violinplot(y=mall['Spending Score (1-100)'],ax=axes[2])
axes[2].set_title('Spending Score (1-100)')
plt.show()
```



There is an outlier in Annual Income (k$) field but Income & Spending Score(1-100) has no outliers

We use Percentile Capping (Winsorization) for outliers handling

In [26]:

```
Q3 = mall['Annual Income (k$)'].quantile(0.99)
Q1 = mall['Annual Income (k$)'].quantile(0.01)
mall['Annual Income (k$)'][mall['Annual Income (k$)']<=Q1]=Q1
mall['Annual Income (k$)'][mall['Annual Income (k$)']>=Q3]=Q3
```

In [27]:

```
# Data After Outlier Treatment
mall.describe()
```

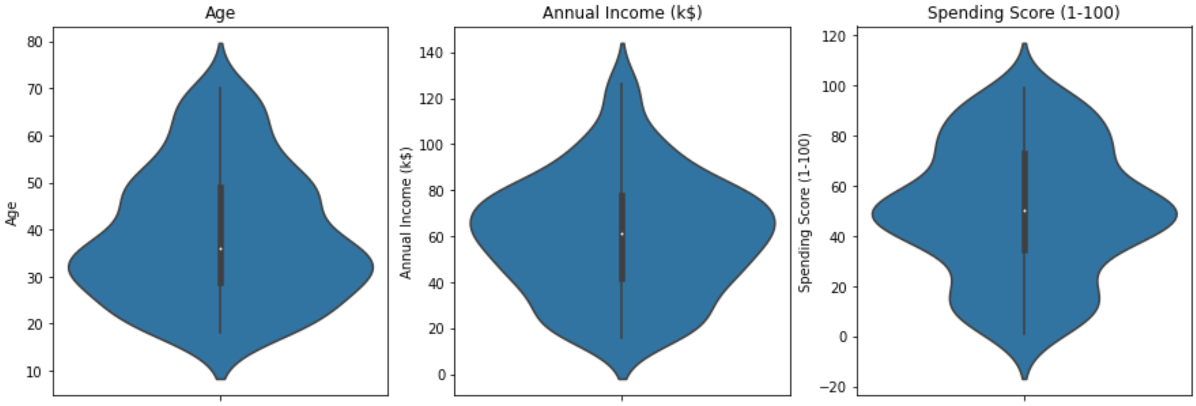
Out[27]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.461000	50.200000
std	57.879185	13.969007	25.949731	25.823522
min	1.000000	18.000000	15.990000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	126.110000	99.000000

In [28]:

```
f, axes = plt.subplots(1,3, figsize=(15,5))
s=sns.violinplot(y=mall.Age,ax=axes[0])
axes[0].set_title('Age')
s=sns.violinplot(y=mall['Annual Income (k$)'],ax=axes[1])
axes[1].set_title('Annual Income (k$)')
s=sns.violinplot(y=mall['Spending Score (1-100)'],ax=axes[2])
axes[2].set_title('Spending Score (1-100)')
plt.show()
```



In [29]:

```
# Dropping CustomerID,Gender field to form cluster

mall_c = mall.drop(['CustomerID','Gender'],axis=1,inplace=True)
```

In [30]:

```
mall.head()
```

Out[30]:

	Age	Annual Income (k$)	Spending Score (1-100)
0	19	15.99	39
1	21	15.99	81

	Age	Annual Income (k$)	Spending Score (1-100)
2	20	16.00	6
3	23	16.00	77
4	31	17.00	40

Conclusion:

In conclusion , customer segmentation through loading and preprocessing of customer data is a critical step in understanding and effectively targeting your customer base. By organizing and analyzing this data, businesses can identify distinct customer groups, tailor their marketing strategies and improve overall customer experiences. This process empowers companies to make data-driven decisions and enhance customer satisfaction, ultimately