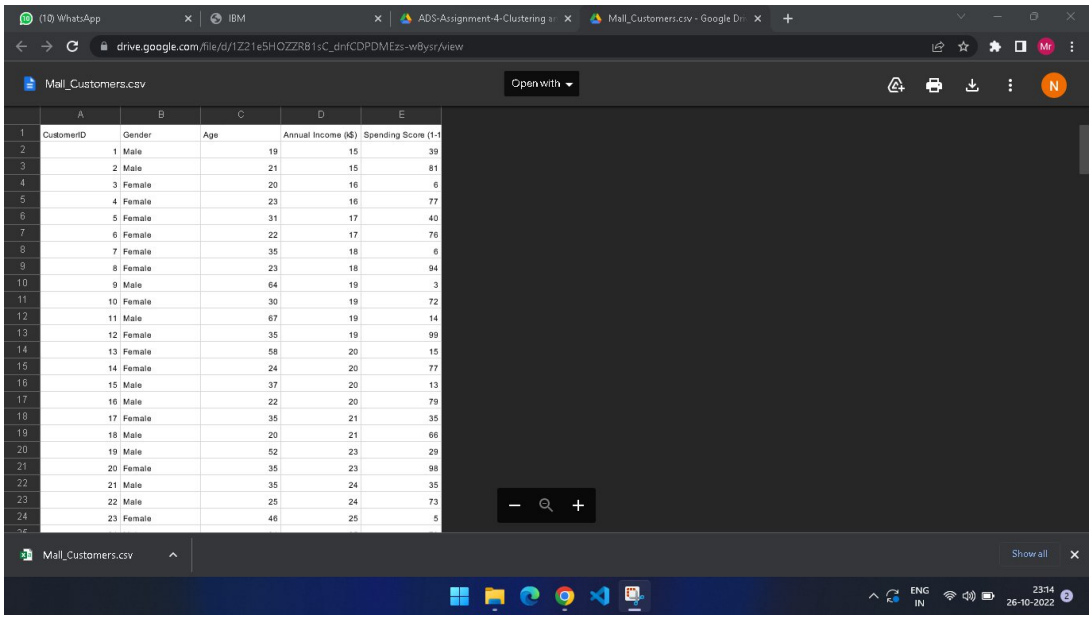
Assignment - 4
Clustering And Classification

Assignment Date	27 October 2022
Project name	University Admit Eligibility Predictor
Team ID	PNT2022TMID42746
Maximum Marks	2 Marks

Question-1:

Download the dataset: Dataset



Question-2:

Load the dataset into the tool

```
In [1]:  1 import pandas
```

```
In [7]:  1 import pandas as pd
         2 df=pd.read_csv("Mall_Customers.csv")
         3 df.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 [ ]:  1
```

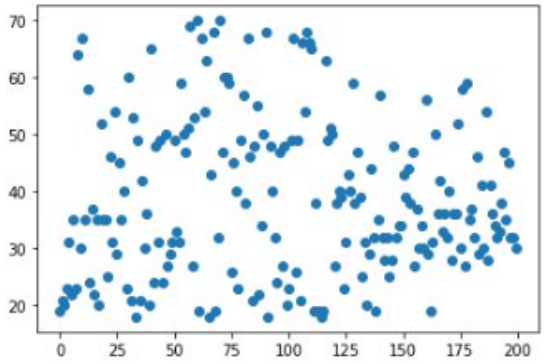
Question-3:

Perform Below Visualizations.

- Univariate analysis

Univariate Analysis

In [8]:
```
1 import matplotlib.pyplot as plt
2 plt.scatter(df.index,df["Age"])
3 plt.show()
```

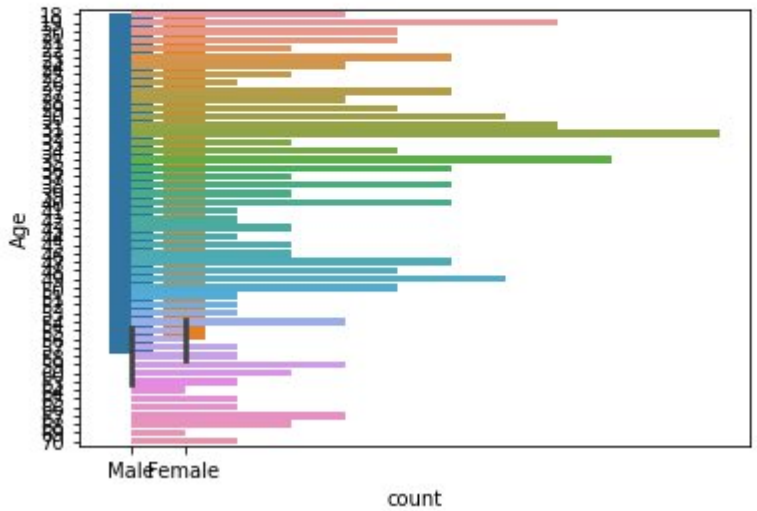


In [ ]:
```
1
```

- Bi-variate analysis

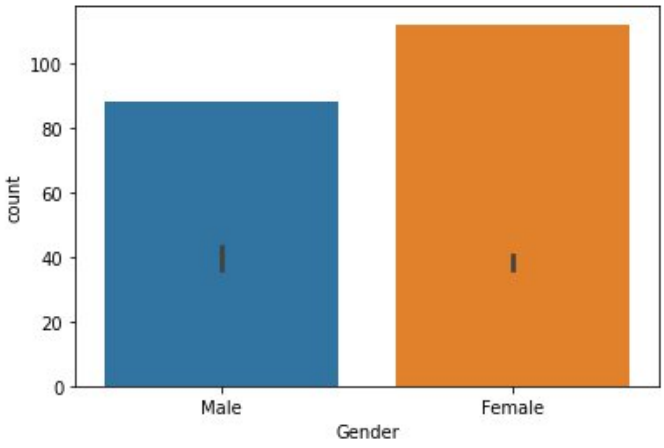
[10]:
```
1 import seaborn as sns
2 sns.barplot(x="Gender" , y="Age", data=df)
3 sns.countplot( y="Age", data=df)
```

[10]: <AxesSubplot:xlabel='count', ylabel='Age'>



In [11]:
```
import seaborn as sns
sns.barplot(x="Gender" , y="Age", data=df)
sns.countplot(x="Gender" , data=df)
```

Out[11]: <AxesSubplot:xlabel='Gender', ylabel='count'>

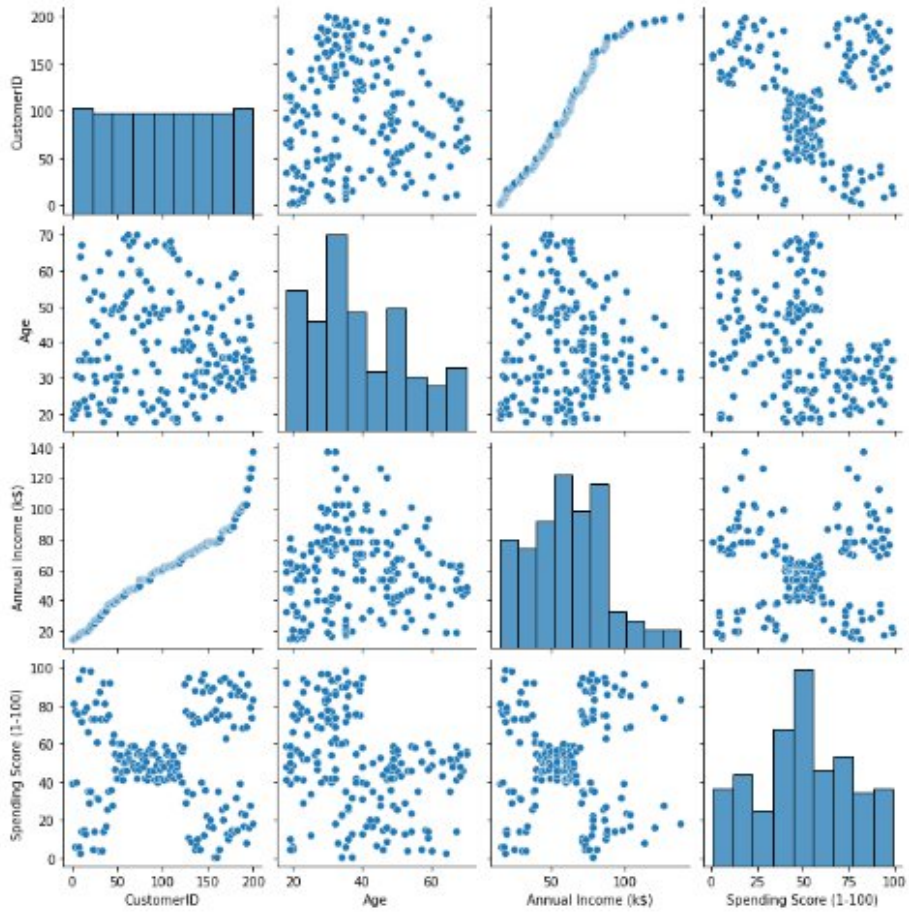


- Multi-variate analysis

Multivariate Analysis

In [14]:
```
import seaborn as sns
sns.pairplot(df)
```

Out[14]: <seaborn.axisgrid.PairGrid at 0x20a184fefd0>



Question-4:

Perform descriptive statistics on the dataset.

```
In [17]: 1 df.mean()
```

```
C:\Users\CYBER\AppData\Local\Temp\ipykernel_8492\3698961737.py:1: FutureWarning: Dropping of nuisance columns in DataFrame
reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid col
umns before calling the reduction.
  df.mean()
```

```
Out[17]: CustomerID               100.50
         Age                       38.85
         Annual Income (k$)        60.56
         Spending Score (1-100)    50.20
         dtype: float64
```

```
In [18]: 1 df.mode()
```

Out[18]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Female	32.0	54.0	42.0
1	2	NaN	NaN	78.0	NaN
2	3	NaN	NaN	NaN	NaN
3	4	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN
...	...	...	...	...	...
195	196	NaN	NaN	NaN	NaN
196	197	NaN	NaN	NaN	NaN
197	198	NaN	NaN	NaN	NaN
198	199	NaN	NaN	NaN	NaN
199	200	NaN	NaN	NaN	NaN

200 rows × 5 columns

```
In [20]: 1 df.median()
```

```
C:\Users\CYBER\AppData\Local\Temp\ipykernel_8492\530051474.py:1: FutureWarning: Dropping of nuisance columns in DataFrame
reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid col
umns before calling the reduction.
  df.median()
```

```
Out[20]: CustomerID               100.5
         Age                       36.0
         Annual Income (k$)        61.5
         Spending Score (1-100)    50.0
         dtype: float64
```

```
In [22]: 1 df.kurt()
```

```
C:\Users\CYBER\AppData\Local\Temp\ipykernel_8492\1257127604.py:1: FutureWarning: Dropping of nuisance columns
reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select on
umns before calling the reduction.
  df.kurt()
```

```
Out[22]: CustomerID              -1.200000
         Age                     -0.671573
         Annual Income (k$)      -0.098487
         Spending Score (1-100)  -0.826629
         dtype: float64
```

```
In [23]:
1 qu=df["Age"].quantile(q=[0.75,0.25])
2 qu
```

```
Out[23]: 0.75    49.00
         0.25    28.75
         Name: Age, dtype: float64
```

```
In [21]:
1 df.skew()
```

```
C:\Users\CYBER\AppData\Local\Temp\ipykernel_8492\1665899112.py:1: Futu
reductions (with 'numeric_only=None') is deprecated; in a future versi
umns before calling the reduction.
  df.skew()
```

```
Out[21]: CustomerID               0.000000
         Age                      0.485569
         Annual Income (k$)       0.321843
         Spending Score (1-100)  -0.047220
         dtype: float64
```

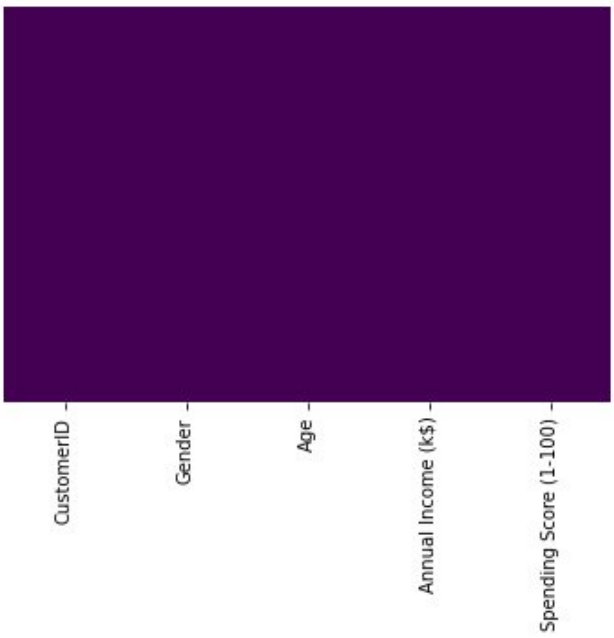
Question-5:

Check for Missing values and deal with them.

Missing values

```
In [29]:
1 sns.heatmap(df.isnull(),yticklabels=False, cbar=False, cmap="viridis")
```

```
Out[29]: <AxesSubplot:>
```



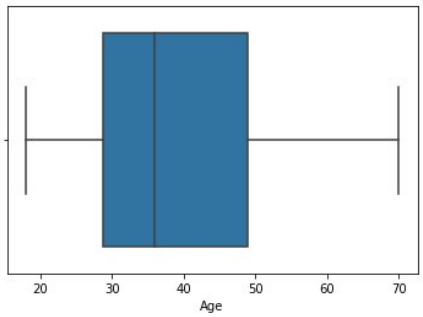
Question-6:

Find the outliers and replace the outliers

```
[31]: sns.boxplot(df["Age"])
```

```
C:\Users\CYBER\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a ke
ord arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
t[31]: <AxesSubplot:xlabel='Age'>
```




```
In [34]:
Q1=df.Age.quantile(0.25)
Q2=df.Age.quantile(0.75)
IQR=Q2-Q1
IQR
```

```
Out[34]: 20.25
```

```
In [38]:
df=df[~((df.Age<(Q1-1.5*IQR)) | (df.Age>(Q2+1.5*IQR)))]
df
```

Out[38]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

Question-7:

Check for Categorical columns and perform encoding.

Encoding

In [39]:
```
df["Gender"].replace({"Female":1, "Male":0},inplace=True)
df.head()
```

Out[39]:

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

Question-8:

Scaling the data

```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
X=std.fit_transform(X)
X
```

```
[-1.13750203, -1.62449091,  1.70038436],
[ 1.80493225, -1.58632148, -1.83237767],
[-0.6351352 , -1.58632148,  0.84631002],
[ 2.02023231, -1.58632148, -1.4053405 ],
[-0.27630176, -1.58632148,  1.89449216],
[ 1.37433211, -1.54815205, -1.36651894],
[-1.06573534, -1.54815205,  1.04041783],
[-0.13276838, -1.54815205, -1.44416206],
[-1.20926872, -1.54815205,  1.11806095],
[-0.27630176, -1.50998262, -0.59008772],
[-1.3528021 , -1.50998262,  0.61338066],
[ 0.94373197, -1.43364376, -0.82301709],
[-0.27630176, -1.43364376,  1.8556706 ],
[-0.27630176, -1.39547433, -0.59008772],
[-0.99396865, -1.39547433,  0.88513158],
[ 0.51313183, -1.3573049 , -1.75473454],
[-0.56336851, -1.3573049 ,  0.88513158],
[ 1.08726535, -1.24279661, -1.4053405 ],
[-0.70690189, -1.24279661,  1.23452563],
[ 0.44136514, -1.24279661, -0.7065524 ],
```

Question-9:

Perform any of the clustering algorithms

```
from sklearn.cluster import MeanShift
clus = MeanShift(bandwidth=2).fit(X)
```

```
from sklearn.cluster import KMeans
kmeans=KMeans(n_clusters=3,random_state=42)
labels=kmeans.fit_predict(X)
```

Question-10:

Add the cluster data with the primary dataset

```
clus.labels_
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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], dtype=int64)
```

Question-11:

Split the data into dependent and independent variables.

```
X=df.iloc[:,2:5].values
```

```
y=df.iloc[:,4].values
y
```

Question-12:

Split the data into training and testing

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

Question-13:

Build the Model

```
from sklearn.linear_model import LinearRegression
r=LinearRegression()
r.fit(x_train,y_train)
```

```
LinearRegression()
```

Question-14:

Train the Model

```
x_train
```

```
array([[-1.3528021 ,  0.4748277 , -1.75473454],
       [ 0.29783176, -0.47940803, -0.00776431],
       [ 0.44136514, -1.24279661, -0.7065524 ],
       [-1.42456879,  0.13130284, -0.16305055],
       [-0.20453507,  1.00919971, -0.90066021],
       [ 1.08726535, -0.51757746,  0.34162973],
       [ 1.80493225, -1.58632148, -1.83237767],
       [-0.92220196, -0.25039146,  0.14752193],
       [ 0.87196528,  0.24581112, -0.27951524],
       [-0.49160182,  0.58933599,  1.42863343],
       [ 0.58489852, -0.4412386 , -0.3183368 ],
       [-1.13750203,  0.36031941, -0.82301709],
       [ 0.15429838,  1.46723286, -0.43480148],
       [-0.85043527, -0.02137488, -0.00776431],
       [-0.34806844,  0.66567484,  1.54509812],
       [ 1.08726535, -1.24279661, -1.4053405 ],
       [ 1.51786549, -1.16645776, -1.7935561 ],
       [ 1.23079873,  0.70384427, -0.59008772],
       [ 1.87669894, -0.86110232, -0.59008772],
```

```
y_train
```

```
array([ 5, 50, 32, 46, 27, 59,  3, 54, 43, 87, 42, 29, 39, 50, 90, 14,  4,
       35, 35, 50, 75, 47, 86, 98, 76, 49, 45, 93, 60, 58, 72, 55, 73, 48,
       12, 46, 48, 13, 61, 93, 68, 55, 10, 79, 43, 52,  6, 46,  7, 74, 61,
       14, 16, 56, 28, 14, 86, 35, 40, 42,  1, 69, 52, 39, 42, 52, 52, 51,
       95, 92, 47, 42, 42, 46, 77, 83, 73, 24, 88, 35, 79, 52, 56, 54, 41,
       53, 77, 43, 26, 18,  6, 59, 57, 40, 89, 75,  1, 41, 83, 99, 57, 59,
       81, 46, 59, 81, 56,  5, 36, 42, 34, 92, 66, 26, 71, 60, 78, 11, 14,
       31, 48, 88, 73, 20, 95, 15,  4, 40, 63, 74, 87, 49, 41, 42, 50, 22,
       91, 49, 48, 82, 75, 55, 17, 13, 23, 75, 51,  5, 60, 55, 55, 17, 73,
       72, 55, 48,  8, 59, 47, 10], dtype=int64)
```

Question-15:

Test the Model

```
x_test
```

```
array([[ 0.94373197, -1.43364376, -0.82301709],
       [ 0.08253169,  1.00919971, -1.44416206],
       [ 1.08726535,  0.09313341, -0.16305055],
       [ 0.65666521,  0.01679455, -0.3183368 ],
       [-0.85043527,  1.04736914,  0.72984534],
       [ 0.51313183,  1.42906343, -1.36651894],
       [-1.20926872, -1.66266033,  1.00159627],
       [ 0.65666521,  0.62750542, -0.55126616],
       [ 1.37433211, -1.54815205, -1.36651894],
       [ 0.36959845,  0.66567484, -1.17241113],
       [-1.42456879, -0.55574689,  0.18634349],
       [-0.56336851,  0.36031941,  1.04041783],
       [-0.13276838,  1.390894  , -0.7065524 ],
       [ 0.58489852,  0.66567484, -1.32769738],
       [ 1.30256542, -0.25039146,  0.03105725],
       [-1.13750203, -1.62449091,  1.70038436],
       [-1.49633548, -1.05194947,  1.62274124],
       [ 0.58489852,  0.39848884, -1.5994483 ],
       [-0.6351352 , -1.01378004,  0.88513158],
       [ 1.4460988 , -0.25039146, -0.12422899],
       [-0.70690189,  1.42906343,  1.46745499],
       [-0.77866858,  0.62750542,  1.81684904],
       [-1.06573534, -0.82293289,  0.5745591 ],
       [-0.6351352 ,  0.66567484,  0.88513158],
       [ 2.23553238, -0.55574689,  0.22516505],
       [ 0.010765  ,  0.32214998,  1.58391968],
       [-0.27630176,  1.23821628,  1.54509812],
```

```
y_test
```

```
array([29, 13, 46, 42, 69, 15, 76, 36, 15, 20, 55, 77, 32, 16, 51, 94, 92,
       9, 73, 47, 88, 97, 65, 73, 56, 91, 90, 97, 58, 28, 35, 41, 17, 54,
       5, 85, 75, 40, 44, 50], dtype=int64)
```

Question-16:

Measure the performance using Evaluation Metrics.

```
from sklearn.metrics import silhouette_score
acc=silhouette_score(X,labels)
print(acc)
```

```
0.357793388710272
```