Assignment-4

PythonProgramming

StudentName	Vasanthkumar D
MaximumMarks	2 Marks

Question-1:

Downloadthedataset:Dataset

Solution:

https://drive.google.com/file/d/1Z21e5HOZZR81sC_dnfCDPDMEzs-w8ysr/view

Question-2:

Loadthe dataset.

Solution:

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

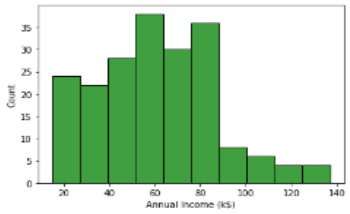
Question-3:

PerformBelowVisualizations.

- UnivariateAnalysis

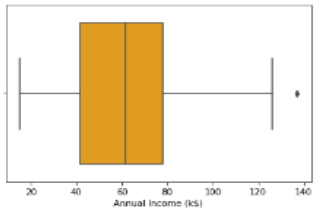
Histplot

```
sns.histplot(data['Annual Income (k$)'], color="green")
```



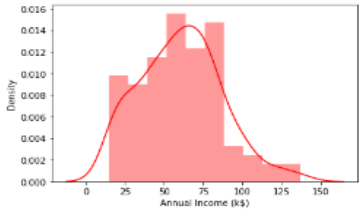
Box Plot

```
sns.boxplot(data['Annual Income (k$)'], color="orange")
```



Dist Plot

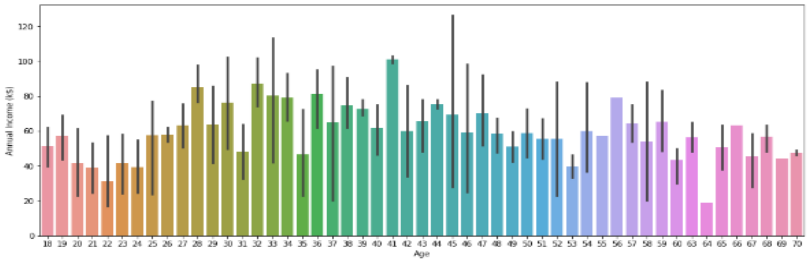
```
sns.distplot(data['Annual Income (k$)'], color="red")
```



- Bi - VariateAnalysis

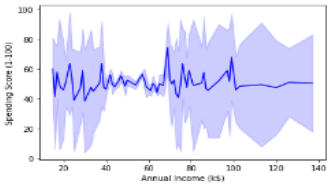
Barplot

```
plt.figure(figsize=(16,6))
sns.barplot(data['Age'],data['Annual Income (k$)'])
```



Lineplot

```
sns.lineplot(data['Annual Income (k$)'], data['Spending Score (1-100)'], color="blue")
```

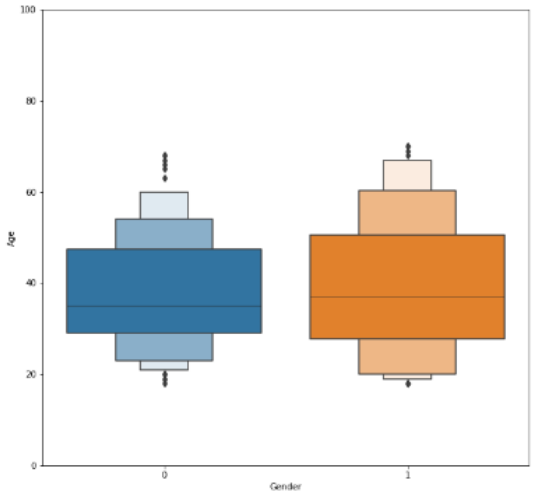


Scatter plot (Age vs Spending Score)

```
sns.scatterplot(data['Spending Score (1-100)'], data['Age'], hue = data['Gender'])
```

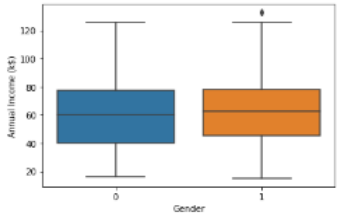
Gender vs Age Distribution

```
temp = pd.concat([data['Age'], data['Gender']], axis=1)
f, ax = plt.subplots(figsize=(10,10))
fig = sns.boxenplot(x='Gender', y="Age", data=data)
fig.axis(ymin=0, ymax=100);
```



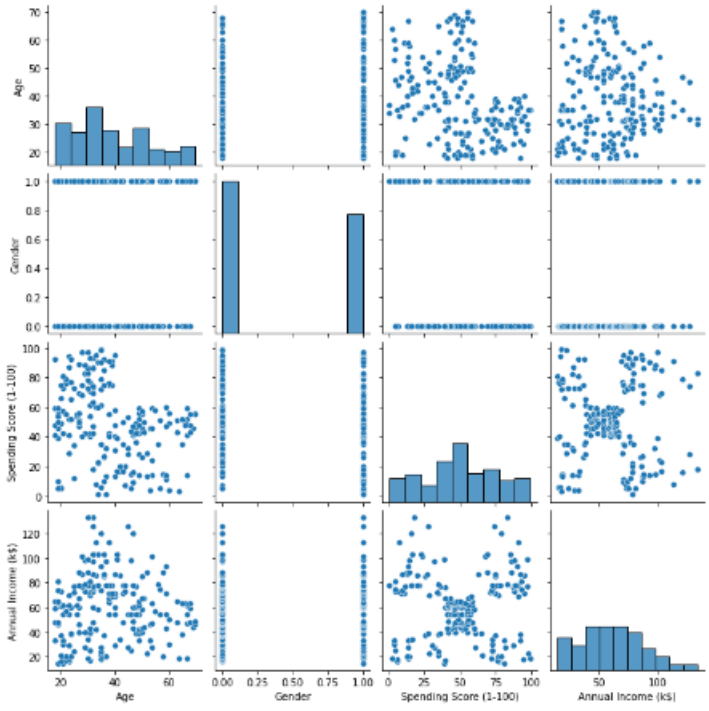
Annual Income vs Gender Countplot

```
sns.boxplot(x=data['Gender'],y=data['Annual Income (k$)'])
```



- Multi-VariateAnalysis

```
sns.pairplot(data=data[["Age", "Gender","Spending Score (1-100)","Annual Income (k$)"]])
```



```
sns.heatmap(data.corr(),annot=True)
```

Question-4:

Perform descriptive statistics on the dataset.

Solution:

```
data.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

```
data.info
```

```
data.shape
```

```
(200, 5)
```

Question-5:

Check for Missing values and deal with them.

Solution:

```
data.isnull().any() #Inference: The dataset has no null values
```

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

```
data.drop('CustomerID',axis=1,inplace=True)
data.head()
```

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

Question-6:

Find the outliers and replace the outliers.

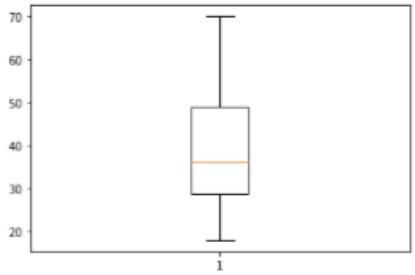
Solution:

```
for i in data:
    if data[i].dtype=='int64':
        q1=data[i].quantile(0.25)
        q3=data[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        data[i]=np.where(data[i] >upper, upper, data[i])
        data[i]=np.where(data[i] <lower, lower, data[i])
```

After removing outliers, boxplot will be like

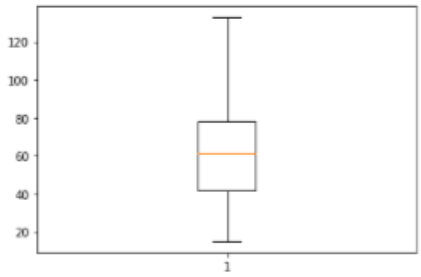
```
plt.boxplot(data['Age'])
```

```
{'whiskers': [,
  ],
 'caps': [,
  ],
 'boxes': [],
 'medians': [],
 'fliers': [],
 'means': []}
```



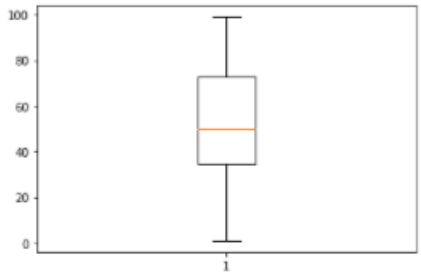

```
plt.boxplot(data['Annual Income (k$)'])
```

```
{'whiskers': [,
  ],
 'caps': [,
  ],
 'boxes': [],
 'medians': [],
 'fliers': [],
 'means': []}
```




```
plt.boxplot(data['Spending Score (1-100)'])
```

```
{'whiskers': [,
  ],
 'caps': [,
  ],
 'boxes': [],
 'medians': [],
 'fliers': [],
 'means': []}
```



Question-7:

Check for Categorical columns and perform encoding.

Solution:

```
from sklearn.preprocessing import LabelEncoder
l_en = LabelEncoder()
```

```
data['Gender'] = l_en.fit_transform(data['Gender'])
data.head()
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19.0	15.0	39.0
1	1	21.0	15.0	81.0
2	0	20.0	16.0	6.0
3	0	23.0	16.0	77.0
4	0	31.0	17.0	40.0

Question-8:

Scaling the data.

Solution:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
data_scaled[0:5]
```

```
array([[1.        , 0.01923077, 0.        , 0.3877551 ],
       [1.        , 0.05769231, 0.        , 0.81632653],
       [0.        , 0.03846154, 0.00849257, 0.05102041],
       [0.        , 0.09615385, 0.00849257, 0.7755102 ],
       [0.        , 0.25      , 0.01698514, 0.39795918]])
```

Question-9:

Perform any of the clustering algorithms

Solution:

```
from sklearn.cluster import KMeans
km = KMeans()
res = km.fit_predict(data_scaled)
res
```

```
array([2, 2, 1, 1, 1, 1, 5, 1, 0, 1, 0, 1, 5, 1, 4, 2, 1, 2, 0, 1, 2, 2,
       5, 2, 5, 2, 5, 2, 5, 1, 0, 1, 0, 2, 5, 1, 5, 1, 5, 1, 5, 2, 0, 1,
       5, 1, 5, 1, 1, 1, 5, 2, 1, 0, 5, 0, 5, 0, 1, 0, 0, 2, 5, 5, 0, 2,
       5, 5, 2, 1, 0, 5, 5, 5, 0, 2, 5, 2, 1, 5, 0, 2, 0, 5, 1, 0, 5, 1,
       1, 5, 5, 2, 0, 5, 1, 2, 5, 1, 0, 2, 1, 5, 0, 2, 0, 1, 5, 0, 0, 0,
       0, 1, 5, 2, 1, 1, 5, 5, 5, 5, 2, 5, 6, 7, 1, 6, 4, 7, 0, 7, 4, 7,
       1, 6, 4, 6, 3, 7, 4, 6, 3, 7, 1, 6, 4, 7, 4, 6, 3, 7, 4, 7, 3, 6,
       3, 6, 4, 6, 4, 6, 5, 6, 4, 6, 4, 6, 4, 6, 3, 7, 4, 7, 4, 7, 3, 6,
       4, 7, 4, 7, 3, 6, 4, 6, 3, 7, 3, 7, 3, 6, 3, 6, 4, 6, 3, 6, 3, 7,
       4, 7], dtype=int32)
```

```
data1 = pd.DataFrame(data_scaled, columns = data.columns)
data1.head()
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.0	0.019231	0.000000	0.387755
1	1.0	0.057692	0.000000	0.816327
2	0.0	0.038462	0.008493	0.051020
3	0.0	0.096154	0.008493	0.775510
4	0.0	0.250000	0.016985	0.397959

```
data1['kclus']  = pd.Series(res)
data1.head()
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	kclus
0	1.0	0.019231	0.000000	0.387755	2
1	1.0	0.057692	0.000000	0.816327	2
2	0.0	0.038462	0.008493	0.051020	1
3	0.0	0.096154	0.008493	0.775510	1
4	0.0	0.250000	0.016985	0.397959	1

```
data1['kclus'].unique()
```

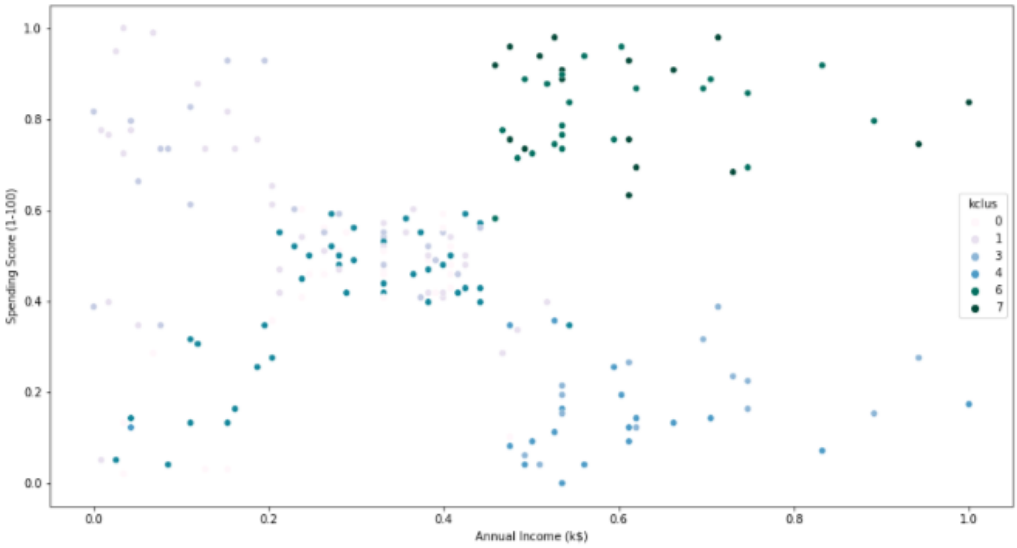
```
array([2, 1, 5, 0, 4, 6, 7, 3], dtype=int32)
```

```
data1['kclus'].value_counts()
```

```
5    39
1    37
0    26
2    24
6    22
4    20
7    18
3    14
Name: kclus, dtype: int64
```

```
import matplotlib.pyplot as plt

fig,ax = plt.subplots(figsize=(15,8))
sns.scatterplot(x=data1['Annual Income (k$)'],
                y=data1['Spending Score (1-100)'],
                hue=data1['kclus'],
                palette='PuBuGn')
plt.show()
```




```
ind = data1.iloc[:,0:4]
ind.head()
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.0	0.019231	0.000000	0.387755
1	1.0	0.057692	0.000000	0.816327
2	0.0	0.038462	0.008493	0.051020
3	0.0	0.096154	0.008493	0.775510
4	0.0	0.250000	0.016985	0.397959

```
dep = data1.iloc[:,4:]
dep.head()
```

	kclus
0	2
1	2
2	1
3	1
4	1

Question-10:

Split the data into training and testing

Solution:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(ind,dep,test_size=0.3,random_state=1)
x_train.head()
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
116	0.0	0.865385	0.424628	0.428571
67	0.0	0.961538	0.280255	0.479592
78	0.0	0.096154	0.331210	0.520408
42	1.0	0.576923	0.203822	0.357143
17	1.0	0.038462	0.050955	0.663265

```
x_test.head()
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
58	0.0	0.173077	0.263270	0.510204
40	0.0	0.903846	0.195329	0.346939
34	0.0	0.596154	0.152866	0.132653
102	1.0	0.942308	0.399151	0.591837
184	0.0	0.442308	0.713376	0.387755

```
y_train.head()
```

	kclus
116	5
67	5
78	1
42	0
17	2

```
y_test.head()
```

	kclus
58	1
40	5
34	5
102	0
184	3

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

```
LinearRegression()
```

```
pred_test = lr.predict(x_test)
pred_test[0:5]
```

```
array([[3.02305666],
       [2.86200206],
       [1.8181892 ],
       [3.65694382],
       [5.20753531]])
```

Question-11:

Measure the performance using Metrics.

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.metrics import accuracy_score
mse = mean_squared_error(pred_test,y_test)
print("The Mean squared error is: ", mse)
rmse = np.sqrt(mse)
print("The Root mean squared error is: ", rmse)
mae = mean_absolute_error(pred_test,y_test)
print("The Mean absolute error is: ", mae)
acc = lr.score(x_test,y_test)
print("The accuracy is: ", acc)
```

```
The Mean squared error is:  4.129095307017881
The Root mean squared error is:  2.0320175459424266
The Mean absolute error is:  1.773889224271428
The accuracy is:  0.23922702772586257
```