ASSIGNMENT 3.1

##CATEGORICAL DATA

In [1]:

```
import pandas as pd
import numpy as np
```

In [4]:

```
dataset = pd.read_csv("mall_customer.csv")
```

In [5]:

```
dataset
```

Out[5]:

	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

In [6]:

```
dataset.shape
```

Out[6]:

```
(200, 5)
```

In [8]:

```
dataset.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]:

```
dataset.mean()
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_9284/1799472221.py:1: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  dataset.mean()
```

Out[10]:

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

In [12]:

```
dataset.loc[:,'Age'].mean()
```

Out[12]:

```
38.85
```

In [13]:

```
dataset.loc[:,'Annual Income (k$)'].mean()
```

Out[13]:

```
60.56
```

In [14]:

```
# calculate men of rows
dataset.mean(axis =1)[0:5]
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_9284/3829306538.py:2: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  dataset.mean(axis =1)[0:5]
```

Out[14]:

```
0    18.50
1    29.75
2    11.25
3    30.00
4    23.25
dtype: float64
```

In [15]:

```
# median represents the 50th percentile or the middle value

dataset.median()
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_9284/3868436747.py:3: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  dataset.median()
```

Out[15]:

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

In [17]:
```
# median of particular variable
dataset.loc[:,'Age'].median()
```

Out[17]:
```
36.0
```

In [19]:
```
# mode represents most  recently accessed
dataset.mode(axis=0)
```

Out[19]:

	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]:
```
dataset.mode()
```

Out[20]:

	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 [21]:

```
# measure of dispersion

# standard deviation

dataset.loc[:,'Age'].std()
```

Out[21]:

```
13.969007331558883
```

In [22]:

```
# standard deviation of first 5 rows
dataset.std(axis =1)[0:5]
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_9284/4071516552.py:2: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  dataset.std(axis =1)[0:5]
```

Out[22]:

```
0    15.695010
1    35.074920
2     8.057088
3    32.300671
4    15.413738
dtype: float64
```

In [23]:

```
# measure varience
dataset.var()
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_9284/2383837622.py:2: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  dataset.var()
```

Out[23]:

```
CustomerID                3350.000000
Age                        195.133166
Annual Income (k$)         689.835578
Spending Score (1-100)     666.854271
dtype: float64
```

In [24]:

```
from scipy.stats import iqr
```

In [25]:

```
iqr(dataset['Age'])
```

Out[25]:

```
20.25
```

In [26]:

```
dataset.skew()
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_9284/4231230252.py:1: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only
valid columns before calling the reduction.
  dataset.skew()
```

Out[26]:

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

In [28]:

```
# describe all the staistics
dataset.describe()
```

Out[28]:

	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 [30]:

```
dataset.describe(include='all')
```

Out[30]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200	200.000000	200.000000	200.000000
unique	NaN	2	NaN	NaN	NaN
top	NaN	Female	NaN	NaN	NaN
freq	NaN	112	NaN	NaN	NaN
mean	100.500000	NaN	38.850000	60.560000	50.200000
std	57.879185	NaN	13.969007	26.264721	25.823522
min	1.000000	NaN	18.000000	15.000000	1.000000
25%	50.750000	NaN	28.750000	41.500000	34.750000
50%	100.500000	NaN	36.000000	61.500000	50.000000
75%	150.250000	NaN	49.000000	78.000000	73.000000
max	200.000000	NaN	70.000000	137.000000	99.000000

In [32]:
```
# prepare groupby

grouped = dataset.groupby('Age')

grouped
```

Out[32]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000026429193B80>
```

In [34]:
```
grouped.groups
```

Out[34]:
```
{18: [33, 65, 91, 114], 19: [0, 61, 68, 111, 113, 115, 138, 162], 20: [2, 17, 3
9, 99, 134], 21: [1, 31, 35, 84, 105], 22: [5, 15, 87], 23: [3, 7, 29, 78, 100,
124], 24: [13, 41, 45, 95], 25: [21, 132, 144], 26: [75, 103], 27: [47, 58, 97,
120, 155, 177], 28: [142, 145, 171, 187], 29: [25, 48, 135, 161, 183], 30: [9,
37, 157, 159, 175, 185, 199], 31: [4, 23, 43, 49, 52, 125, 133, 163], 32: [69,
94, 137, 141, 143, 147, 169, 181, 191, 197, 198], 33: [51, 167, 192], 34: [88,
148, 149, 158, 190], 35: [6, 11, 16, 19, 20, 27, 139, 179, 195], 36: [38, 165,
168, 172, 173, 189], 37: [14, 156, 180], 38: [81, 112, 121, 129, 153, 193], 39:
[123, 131, 151], 40: [28, 77, 93, 122, 127, 170], 41: [184, 188], 42: [36, 16
6], 43: [66, 126, 150], 44: [136, 152], 45: [26, 76, 196], 46: [22, 83, 182], 4
7: [55, 71, 96, 130, 154, 194], 48: [42, 85, 92, 98, 146], 49: [34, 44, 50, 79,
101, 104, 117], 50: [46, 54, 89, 119, 164], 51: [56, 118], 52: [18, 174], 53:
[32, 59], 54: [24, 63, 107, 186], 55: [86], 56: [160], 57: [80, 140], 58: [12,
176], 59: [53, 74, 128, 178], 60: [30, 72, 73], 63: [64, 116], 64: [8], 65: [4
0, 110], 66: [106, 109], 67: [10, 62, 82, 102], 68: [67, 90, 108], 69: [57], 7
0: [60, 70]}
```

In [35]:
```
grouped.size()
```

Out[35]:
```
Age
18     4
19     8
20     5
21     5
22     3
23     6
24     4
25     3
26     2
27     6
28     4
29     5
30     7
31     8
32    11
33     3
34     5
35     9
36     6
37     3
38     6
39     3
40     6
41     2
42     2
43     3
44     2
45     3
46     3
47     6
48     5
49     7
50     5
51     2
52     2
53     2
54     4
55     1
56     1
57     2
58     2
59     4
60     3
63     2
64     1
65     2
66     2
67     4
68     3
69     1
70     2
dtype: int64
```

In [36]:
```
grouped["Age"]
```

Out[36]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x000002642919EC10>
```

In [37]:

```
grouped["Age"].size()
```

Out[37]:

```
Age
18     4
19     8
20     5
21     5
22     3
23     6
24     4
25     3
26     2
27     6
28     4
29     5
30     7
31     8
32    11
33     3
34     5
35     9
36     6
37     3
38     6
39     3
40     6
41     2
42     2
43     3
44     2
45     3
46     3
47     6
48     5
49     7
50     5
51     2
52     2
53     2
54     4
55     1
56     1
57     2
58     2
59     4
60     3
63     2
64     1
65     2
66     2
67     4
68     3
69     1
70     2
Name: Age, dtype: int64
```

In [39]:

```
# counting number of each category by count()
print(dataset.groupby(["Gender"]).count().reset_index())
```

```
   Gender  CustomerID  Age  Annual Income (k$)  Spending Score (1-100)
0  Female         112  112                 112                     112
1    Male          88   88                  88                      88
```

In [40]:

```
print(dataset.groupby(['Gender']).mean().reset_index())
```

```
   Gender  CustomerID        Age  Annual Income (k$)  Spending Score (1-100)
0  Female   97.562500  38.098214           59.250000               51.526786
1    Male  104.238636  39.806818           62.227273               48.511364
```

ASSIGNMENT 3.2

In [1]:

```
import pandas as pd
data = pd.read_csv("iris.csv")
```

In [3]:

```
print('iris-setosa')
setosa = data["Species"]=='iris-setosa'
print(data[setosa].describe())
print('\niris-versicolor')
setosa = data['Species'] == 'iris-versicolor'
print(data[setosa].describe())
print('\niris-virginics')
setosa = data['Species']=='iris-virginica'
print(data[setosa].describe())
```

```
iris-setosa
        Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  0.0            0.0           0.0            0.0           0.0
mean   NaN            NaN           NaN            NaN           NaN
std    NaN            NaN           NaN            NaN           NaN
min    NaN            NaN           NaN            NaN           NaN
25%    NaN            NaN           NaN            NaN           NaN
50%    NaN            NaN           NaN            NaN           NaN
75%    NaN            NaN           NaN            NaN           NaN
max    NaN            NaN           NaN            NaN           NaN

iris-versicolor
        Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  0.0            0.0           0.0            0.0           0.0
mean   NaN            NaN           NaN            NaN           NaN
std    NaN            NaN           NaN            NaN           NaN
min    NaN            NaN           NaN            NaN           NaN
25%    NaN            NaN           NaN            NaN           NaN
50%    NaN            NaN           NaN            NaN           NaN
75%    NaN            NaN           NaN            NaN           NaN
max    NaN            NaN           NaN            NaN           NaN

iris-virginics
        Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  0.0            0.0           0.0            0.0           0.0
mean   NaN            NaN           NaN            NaN           NaN
std    NaN            NaN           NaN            NaN           NaN
min    NaN            NaN           NaN            NaN           NaN
25%    NaN            NaN           NaN            NaN           NaN
50%    NaN            NaN           NaN            NaN           NaN
75%    NaN            NaN           NaN            NaN           NaN
max    NaN            NaN           NaN            NaN           NaN
```

In [15]:

```
import pandas as pd
import numpy as np
csv_url='https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data
```

In [16]:

```
iris = pd.read_csv(csv_url,header=None)
```

In [18]:

```
col_names=['Sepal_length',"Sepal_Width","Petal_Length","Petal_Width","Species"]
```

In [19]:

```
iris = pd.read_csv(csv_url,names=col_names)
print(iris)
```

```
     Sepal_length  Sepal_Width  Petal_Length  Petal_Width         Species
0             5.1          3.5           1.4          0.2     Iris-setosa
1             4.9          3.0           1.4          0.2     Iris-setosa
2             4.7          3.2           1.3          0.2     Iris-setosa
3             4.6          3.1           1.5          0.2     Iris-setosa
4             5.0          3.6           1.4          0.2     Iris-setosa
..            ...          ...           ...          ...             ...
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]
```

In [20]:

```
irisSet = (iris['Species']=='Iris-setosa')
print('Iris-setosa')
print(iris[irisSet].describe())

irisVer =(iris['Species']=='Iris-versicolor')
print('Iris-versicolor')
print(iris[irisVer].describe())

irisVir = (iris['Species']=='Iris-virginica')
print('Iris-virginica')
print(iris[irisVir].describe())
```

```
Iris-setosa
       Sepal_length  Sepal_Width  Petal_Length  Petal_Width
count      50.00000    50.000000     50.000000     50.00000
mean        5.00600     3.418000      1.464000      0.24400
std         0.35249     0.381024      0.173511      0.10721
min         4.30000     2.300000      1.000000      0.10000
25%         4.80000     3.125000      1.400000      0.20000
50%         5.00000     3.400000      1.500000      0.20000
75%         5.20000     3.675000      1.575000      0.30000
max         5.80000     4.400000      1.900000      0.60000
Iris-versicolor
       Sepal_length  Sepal_Width  Petal_Length  Petal_Width
count     50.000000    50.000000     50.000000    50.000000
mean       5.936000     2.770000      4.260000     1.326000
std        0.516171     0.313798      0.469911     0.197753
min        4.900000     2.000000      3.000000     1.000000
25%        5.600000     2.525000      4.000000     1.200000
50%        5.900000     2.800000      4.350000     1.300000
75%        6.300000     3.000000      4.600000     1.500000
max        7.000000     3.400000      5.100000     1.800000
Iris-virginica
       Sepal_length  Sepal_Width  Petal_Length  Petal_Width
count      50.00000    50.000000     50.000000     50.00000
mean        6.58800     2.974000      5.552000      2.02600
std         0.63588     0.322497      0.551895      0.27465
min         4.90000     2.200000      4.500000      1.40000
25%         6.22500     2.800000      5.100000      1.80000
50%         6.50000     3.000000      5.550000      2.00000
75%         6.90000     3.175000      5.875000      2.30000
max         7.90000     3.800000      6.900000      2.50000
```

In [ ]: