In [1]:

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

In [2]:

```python
age=pd.read_csv('age.csv')
age
```

Out[2]:

	CustomerID	Genre	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 [3]:

```python
age.describe()
```

Out[3]:

	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

Mean

In [4]:

```python
age.mean()
```

```
C:\Users\vishal\AppData\Local\Temp/ipykernel_20920/616271996.py:1: FutureWarning: Droppi
ng of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated;
in a future version this will raise TypeError.  Select only valid columns before calling
the reduction.
  age.mean()
```

Out[4]:

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

In [5]:

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

Out[5]:

```
38.85
```

In [6]:

```
age.mean(axis=1)[0:4]
```

```
C:\Users\vishal\AppData\Local\Temp/ipykernel_20920/791093223.py:1: FutureWarning: Droppi
ng of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated;
in a future version this will raise TypeError.  Select only valid columns before calling
the reduction.
  age.mean(axis=1)[0:4]
```

Out[6]:

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

Median

In [7]:

```
age.median()
```

```
C:\Users\vishal\AppData\Local\Temp/ipykernel_20920/4265497832.py:1: FutureWarning: Dropp
ing of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecate
d; in a future version this will raise TypeError.  Select only valid columns before call
ing the reduction.
  age.median()
```

Out[7]:

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

In [8]:

```
age.loc[:,'Age'].median()
```

Out[8]:

```
36.0
```

In [9]:

```
age.median(axis=1)[0:4]
```

```
C:\Users\vishal\AppData\Local\Temp/ipykernel_20920/387372506.py:1: FutureWarning: Droppi
ng of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated;
in a future version this will raise TypeError.  Select only valid columns before calling
```

```
the reduction.
  age.median(axis=1)[0:4]
```

Out[9]:

```
0    17.0
1    18.0
2    11.0
3    19.5
dtype: float64
```

Mode

In [10]:

```
age.mode()
```

Out[10]:

	CustomerID	Genre	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 [11]:

```
age.loc[:,'Age'].mode()
```

Out[11]:

```
0    32
dtype: int64
```

In [12]:

```
age.mode(axis=1)[0:4]
```

```
C:\Users\vishal\anaconda3\lib\site-packages\pandas\core\algorithms.py:969: UserWarning:
Unable to sort modes: '<' not supported between instances of 'str' and 'int'
  warn(f"Unable to sort modes: {err}")
```

Out[12]:

	0	1	2	3	4
0	1	Male	19.0	15.0	39.0
1	2	Male	21.0	15.0	81.0
2	3	Female	20.0	16.0	6.0
3	4	Female	23.0	16.0	77.0

Minimum

In [13]:

```
age.min()
```

Out[13]:

```
CustomerID                     1
Genre                     Female
Age                           18
Annual Income (k$)            15
Spending Score (1-100)         1
dtype: object
```

In [14]:

```
age.loc[:,'Age'].min(skipna= False)
```

Out[14]:

```
18
```

Maximum

In [15]:

```
age.max()
```

Out[15]:

```
CustomerID                 200
Genre                     Male
Age                         70
Annual Income (k$)         137
Spending Score (1-100)      99
dtype: object
```

In [16]:

```
age.loc[:,'Age'].max(skipna = False)
```

Out[16]:

```
70
```

Standard Deviation

In [17]:

```
age.std()
```

```
C:\Users\vishal\AppData\Local\Temp/ipykernel_20920/4118967071.py:1: FutureWarning: Dropp
ing of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecate
d; in a future version this will raise TypeError.  Select only valid columns before call
ing the reduction.
  age.std()
```

Out[17]:

```
CustomerID                57.879185
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

In [18]:

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

Out[18]:

```
13.969007331558883
```

In [19]:

```
age.std(axis=1)[0:4]
```

```
C:\Users\vishal\AppData\Local\Temp/ipykernel_20920/2864792906.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 columns before calling the reduction.
  age.std(axis=1)[0:4]
```

Out[19]:

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

Summary statistics of income grouped by the age groups

In [20]:

```
age.groupby(['Genre'])['Age'].mean()
```

Out[20]:

```
Genre
Female    38.098214
Male      39.806818
Name: Age, dtype: float64
```

In [21]:

```
age_w=age.rename(columns={'Annual Income (k$)':'income'}, inplace=False)
```

In [22]:

```
age_w
```

Out[22]:

	CustomerID	Genre	Age	income	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 [23]:

```
age_w.groupby(['Genre'])['income'].mean()
```

Out[23]:

```
Genre
Female    59.250000
Male      62.227273
Name: income, dtype: float64
```

In [24]:

```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(age[['Genre']]).toarray())
enc_df
```

Out[24]:

	0	1
0	0.0	1.0
1	0.0	1.0
2	1.0	0.0
3	1.0	0.0
4	1.0	0.0
...	...	...
195	1.0	0.0
196	1.0	0.0
197	0.0	1.0
198	0.0	1.0
199	0.0	1.0

200 rows × 2 columns

In [25]:

```
df_encode =age_w.join(enc_df)
df_encode
```

Out[25]:

	CustomerID	Genre	Age	income	Spending Score (1-100)	0	1
0	1	Male	19	15	39	0.0	1.0
1	2	Male	21	15	81	0.0	1.0
2	3	Female	20	16	6	1.0	0.0
3	4	Female	23	16	77	1.0	0.0
4	5	Female	31	17	40	1.0	0.0
...	...	...	...	...	...	...	...
195	196	Female	35	120	79	1.0	0.0
196	197	Female	45	126	28	1.0	0.0
197	198	Male	32	126	74	0.0	1.0
198	199	Male	32	137	18	0.0	1.0
199	200	Male	30	137	83	0.0	1.0

200 rows × 7 columns

Basic Statistical details on iris dataset

In [26]:

```
iris=pd.read_csv('iris.csv')
iris
```

Out[26]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [27]:

```
iris['species'].unique()
```

Out[27]:

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [29]:

```
print("setosa")
setosa = iris['species'] == 'setosa'
print(iris[setosa].describe())
print("versicolor")
versicolor=iris['species'] == 'versicolor'
print(iris[versicolor].describe())
print("virginica")
virginica=iris['species'] == 'virginica'
print(iris[virginica].describe())
```

```
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
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
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 [ ]: