```
#1. mount Google drive
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Mounted at /content/gdrive
```

```
#2. Load datset in g drive

import pandas as pd
df=pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/Datasets/Assign 3 Mall_Customers.csv
df1=df
print(df)
```

```
     CustomerID   Genre   Age Group  Age  Annual Income (k$)  \
0             1    Male        Teen   19                  15
1             2    Male  Middle Age   21                  15
2             3  Female  Middle Age   20                  16
3             4  Female  Middle Age   23                  16
4             5  Female  Middle Age   31                  17
..          ...     ...         ...  ...                 ...
195         196  Female  Middle Age   35                 120
196         197  Female       Elder   45                 126
197         198    Male  Middle Age   32                 126
198         199    Male  Middle Age   32                 137
199         200    Male  Middle Age   30                 137

     Spending Score (1-100)
0                        39
1                        81
2                         6
3                        77
4                        40
..                      ...
195                      79
196                      28
197                      74
198                      18
199                      83

[200 rows x 6 columns]
```

```
#code to see whole dataframe
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more option
    print(df)
```

```
# 1. Import all required python libraries

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

```
df.shape
```

```
(200, 6)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Genre                   200 non-null    object
 2   Age Group               200 non-null    object
 3   Age                     200 non-null    int64 
 4   Annual Income (k$)      200 non-null    int64 
 5   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 9.5+ KB
```

```
#Two of the variables are categorical (labelled as 'object') while the remaining are numerica
```

```
df.isnull().sum()
```

```
CustomerID              0
Genre                   0
Age Group               0
Age                     0
Annual Income (k$)      0
Spending Score (1-100)  0
dtype: int64
```

```
# delete CustomerID from data frame
df.drop(["CustomerID"], axis=1,inplace=True)
```

```
df
```

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

```
df.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
Age                     38.85
Annual Income (k$)      60.56
Spending Score (1-100)  50.20
dtype: float64
```

200 rows × 5 columns

```
#It is also possible to calculate the mean of a particular variable in a data, as shown below
print(df.loc[:,'Age'].mean())
```

```
38.85
```

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

```
60.56
```

```
#In the previous sections, we computed the column-wise mean. It is also possible to calculate
df.mean(axis = 1)[0:5]
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping
```

```
0    24.333333
1    39.000000
2    14.000000
3    38.666667
4    29.333333
dtype: float64
```

Median In simple terms, median represents the 50th percentile, or the middle value of the data, that separates the distribution into two halves. The line of code below prints the median of the numerical variables in the data. The command df.median(axis = 0) will also give the same output.

```
df.median()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
Age                     36.0
Annual Income (k$)      61.5
Spending Score (1-100)  50.0
dtype: float64
```

From the output, we can infer that the median age of the applicants is 36 years, the median annual income is USD 61.5 k$, and the median of spending scores is 50 . If there is a difference between the mean and the median values of these variables, it is because of the distribution of the data.

It is also possible to calculate the median of a particular variable in a data.

```
#to calculate a median of a particular column
print(df.loc[:,'Age'].median())
print(df.loc[:,'Annual Income (k$)'].median())
```

```
36.0
61.5
```

```
#OR
df[["Age", "Annual Income (k$)"]].median()
```

```
Age                   36.0
Annual Income (k$)    61.5
dtype: float64
```

Mode Mode represents the most frequent value of a variable in the data. This is the only central tendency measure that can be used with categorical variables, unlike the mean and the median which can be used only with quantitative data.

```
df.mode()
```

	Genre	Age Group	Age	Annual Income (k$)	Spending Score (1-100)
0	Female	Middle Age	32.0	54	42.0
1	NaN	NaN	NaN	78	NaN

```
#OR
df.mode(axis = 0)
```

	Genre	Age Group	Age	Annual Income (k$)	Spending Score (1-100)
0	Female	Middle Age	32.0	54	42.0
1	NaN	NaN	NaN	78	NaN

Measures of Dispersion In the previous sections, we have discussed the various measures of central tendency. However, as we have seen in the data, the values of these measures differ for many variables. This is because of the extent to which a distribution is stretched or squeezed. In statistics, this is measured by dispersion which is also referred to as variability, scatter, or spread. The most popular measures of dispersion are standard deviation, variance, and the interquartile range.

Standard Deviation Standard deviation is a measure that is used to quantify the amount of variation of a set of data values from its mean. A low standard deviation for a variable indicates that the data points tend to be close to its mean, and vice versa.

```
df.std()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
Age                     13.969007
Annual Income (k$)      26.264721
Spending Score (1-100)  25.823522
dtype: float64
```

It is also possible to calculate the standard deviation of a particular variable, as shown in the first two lines of code below. The third line calculates the standard deviation for the first five rows.

```
print(df.loc[:,'Age'].std())
print(df.loc[:,'Annual Income (k$)'].std())
```

```
13.969007331558883
26.264721165271254
```

```
#calculate the standard deviation of the first five rows
df.std(axis = 1)[0:5]
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping

0    12.858201
1    36.496575
2     7.211103
3    33.381632
```

```
4    11.590226
dtype: float64
```

Variance Variance is another measure of dispersion. It is the square of the standard deviation and the covariance of the random variable with itself. The line of code below prints the variance of all the numerical variables in the dataset. The interpretation of the variance is similar to that of the standard deviation.

```
df.var()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
Age                     195.133166
Annual Income (k$)      689.835578
Spending Score (1-100)  666.854271
dtype: float64
```

Interquartile Range (IQR) The Interquartile Range (IQR) is a measure of statistical dispersion, and is calculated as the difference between the upper quartile (75th percentile) and the lower quartile (25th percentile). The IQR is also a very important measure for identifying outliers and could be visualized using a boxplot.

IQR can be calculated using the iqr() function. The first line of code below imports the 'iqr' function from the scipy.stats module, while the second line prints the IQR for the variable 'Age'.

```
from scipy.stats import iqr
iqr(df['Age'])
```

```
20.25
```

Skewness Another useful statistic is skewness, which is the measure of the symmetry, or lack of it, for a real-valued random variable about its mean. The skewness value can be positive, negative, or undefined. In a perfectly symmetrical distribution, the mean, the median, and the mode will all have the same value. However, the variables in our data are not symmetrical, resulting in different values of the central tendency.

The skewness values can be interpreted in the following manner:

Highly skewed distribution: If the skewness value is less than −1 or greater than +1.

Moderately skewed distribution: If the skewness value is between −1 and −½ or between +½ and +1.

Approximately symmetric distribution: If the skewness value is between −½ and +½.

We can calculate the skewness of the numerical variables using the skew() function, as shown below.

```
print(df.skew())
```

```
Age                     0.485569
Annual Income (k$)      0.321843
Spending Score (1-100)  -0.047220
dtype: float64
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping
  """Entry point for launching an IPython kernel.
```

Putting Everything Together We have learned the measures of central tendency and dispersion, in the previous sections. It is important to analyse these individually, however, because there are certain useful functions in python that can be called upon to find these values. One such important function is the .describe() function that prints the summary statistic of the numerical variables.

```
df.describe()
```

	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000
mean	38.850000	60.560000	50.200000
std	13.969007	26.264721	25.823522
min	18.000000	15.000000	1.000000
25%	28.750000	41.500000	34.750000
50%	36.000000	61.500000	50.000000
75%	49.000000	78.000000	73.000000
max	70.000000	137.000000	99.000000

The above output prints the important summary statistics of all the numerical variables like the mean, median (50%), minimum, and maximum values, along with the standard deviation. We can also calculate the IQR using the 25th and 75th percentile values.

However, the 'describe()' function only prints the statistics for the quantitative or numerical variable. In order to print the similar statistics for all the variables, an additional argument, include='all', needs

to be added, as shown in the line of code below.

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

	Genre	Age Group	Age	Annual Income (k$)	Spending Score (1-100)
count	200	200	200.000000	200.000000	200.000000
unique	2	3	NaN	NaN	NaN
top	Female	Middle Age	NaN	NaN	NaN
freq	112	104	NaN	NaN	NaN
mean	NaN	NaN	38.850000	60.560000	50.200000
std	NaN	NaN	13.969007	26.264721	25.823522
min	NaN	NaN	18.000000	15.000000	1.000000
25%	NaN	NaN	28.750000	41.500000	34.750000
50%	NaN	NaN	36.000000	61.500000	50.000000
75%	NaN	NaN	49.000000	78.000000	73.000000
max	NaN	NaN	70.000000	137.000000	99.000000

#Que: Provide summary statistics of income grouped by age group

```
df2=df.groupby(['Genre', 'Age Group'])
df2.first()
```

		Age	Annual Income (k$)	Spending Score (1-100)
Genre	**Age Group**			
Female	**Elder**	58	20	15
	Middle Age	20	16	6
	Teen	19	63	54
Male	**Elder**	64	19	3
	Middle Age	21	15	81
	Teen	19	15	39

```
# grouping income as per the age groups and calculating its meae,median and mode values using
df3=df.groupby(['Age Group', 'Annual Income (k$)']).mean()
df3
```

		Age	Spending Score (1-100)
Age Group	**Annual Income (k$)**		
Elder	**19**	65.5	8.5
	20	58.0	15.0
	23	52.0	29.0
	25	46.0	5.0
	28	49.5	23.0
...	**...**	...	...
Teen	**63**	19.0	54.0
	64	19.0	46.0
	65	18.5	49.0
	74	19.0	10.0
	81	19.0	5.0

115 rows × 2 columns

```
df4=df.groupby(['Age Group', 'Annual Income (k$)']).median()
df4
```

```
                                    Age  Spending Score (1-100)
#how to find the mode using pandas groupby

df.groupby(['Age Group','Annual Income (k$)']).agg(lambda x:x.value_counts().index[0])
```

		Genre	Age	Spending Score (1-100)
Age Group	**Annual Income (k$)**			
Elder	**19**	Male	64	3
	20	Female	58	15
	23	Male	52	29
	25	Female	46	5
	28	Female	54	14
...	**...**	...	...	...
Teen	**63**	Female	19	54
	64	Male	19	46
	65	Female	18	48
	74	Male	19	10
	81	Male	19	5

115 rows × 3 columns

```
#List that contains a numeric value for each responce to the categorical value
df5=df.groupby(['Age Group', 'Age', 'Annual Income (k$)'])
df5.first()
```

			Genre	Spending Score (1-100)
Age Group	Age	Annual Income (k$)		
Elder	**40**	**29**	Female	31
		54	Male	48
		60	Female	40
		71	Male	95
...	**...**	**...**	...	...
Teen	**19**	**63**	Female	54
		64	Male	46
		65	Female	50
		74	Male	10
		81	Male	5

192 rows × 2 columns