In [1]:
```
#Descriptive Statistics is the building block of data science. Advanced analytics is c

#Measures of central tendency include mean, median, and the mode, while the measures c

#We will cover the topics given below:

#Mean
#Median
#Mode
#Standard Deviation
#Variance
#Interquartile Range
#Skewness
```

In [3]:
```
import pandas as pd
import numpy as np
import statistics as st

# Load the data
df = pd.read_csv("loan_data.csv")
print(df.shape)
print(df.info())
```

```
(614, 14)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Age                513 non-null    float64
 3   Married            611 non-null    object 
 4   Dependents         599 non-null    object 
 5   Education          614 non-null    object 
 6   Self_Employed      582 non-null    object 
 7   ApplicantIncome    614 non-null    int64  
 8   CoapplicantIncome  614 non-null    float64
 9   LoanAmount         592 non-null    float64
 10  Loan_Amount_Term   600 non-null    float64
 11  Credit_History     564 non-null    float64
 12  Property_Area      614 non-null    object 
 13  Loan_Status        614 non-null    object 
dtypes: float64(5), int64(1), object(8)
memory usage: 67.3+ KB
None
```

In [4]:
```
df.mean()
```

```
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/3698961737.py:1: FutureWarning: Droppi
ng 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 ca
lling the reduction.
  df.mean()
```

Out[4]:
```
Age                  32.101365
ApplicantIncome    5403.459283
CoapplicantIncome  1621.245798
LoanAmount          146.412162
Loan_Amount_Term    342.000000
```

```
Credit_History      0.842199
dtype: float64
```

In [5]:
```
print(df.loc[:,'Age'].mean())
print(df.loc[:,'ApplicantIncome'].mean())
```

```
32.10136452241716
5403.459283387622
```

In [6]:
```
df.mean(axis = 1)[0:10]
```

```
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/1983962910.py:1: FutureWarning: Droppi
ng 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 ca
lling the reduction.
  df.mean(axis = 1)[0:10]
```

Out[6]:
```
0    1249.000000
1    1316.000000
2     575.333333
3     908.000000
4    1088.666667
5    1713.500000
6     722.500000
7    1017.166667
8    1017.833333
9    4092.833333
dtype: float64
```

In [7]:
```
df.median(axis=1)
```

```
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/3214761434.py:1: FutureWarning: Droppi
ng 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 ca
lling the reduction.
  df.median(axis=1)
```

Out[7]:
```
0      35.0
1     360.0
2      45.5
3     240.0
4      85.5
      ...
609    48.0
610    45.0
611   246.5
612   116.0
613    79.5
Length: 614, dtype: float64
```

In [8]:
```
df.median()
```

```
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/530051474.py:1: FutureWarning: Droppin
g 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 ca
lling the reduction.
  df.median()
```

Out[8]:
```
Age                 30.0
ApplicantIncome   3812.5
CoapplicantIncome 1188.5
LoanAmount          128.0
Loan_Amount_Term    360.0
```

```
Credit_History          1.0
dtype: float64
```

In [9]:
```
#to compute a median of a some column
print(df.loc[:,'Age'].median())
print(df.loc[:,'ApplicantIncome'].median())

df.median(axis = 1)[0:10]
```

```
30.0
3812.5
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/1048502292.py:5: 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.
  df.median(axis = 1)[0:10]
```

Out[9]:
```
0     35.0
1    360.0
2     45.5
3    240.0
4     85.5
5    313.5
6    227.5
7    259.0
8    264.0
9    354.5
dtype: float64
```

In [10]:
```
df.mode()
```

Out[10]:

	Loan_ID	Gender	Age	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coa
0	LP001002	Male	25.0	Yes	0	Graduate	No	2500.0	
1	LP001003	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	LP001005	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	LP001006	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	LP001008	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	
609	LP002978	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
610	LP002979	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
611	LP002983	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
612	LP002984	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
613	LP002990	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

614 rows × 14 columns

Measures the Dispersion

In [11]:
```
#Measure the Standard deviation
df.std()
```

```
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/1952243819.py:2: FutureWarning: Droppi
ng 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 ca
lling the reduction.
  df.std()
```

Out[11]:
```
Age                  7.732178
ApplicantIncome   6109.041673
CoapplicantIncome 2926.248369
LoanAmount          85.587325
Loan_Amount_Term    65.120410
Credit_History       0.364878
dtype: float64
```

In [12]:
```
print(df.loc[:,'Age'].std())
print(df.loc[:,'ApplicantIncome'].std())

#calculate the standard deviation of the first five rows
df.std(axis = 1)[0:10]
```

```
7.732178229043358
6109.041673387174
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/1761117073.py:5: FutureWarning: Droppi
ng 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 ca
lling the reduction.
  df.std(axis = 1)[0:10]
```

Out[12]:
```
0    2575.928085
1    1921.240355
2    1195.703252
3    1219.011567
4    2409.946528
5    2430.485610
6     974.539224
7    1373.763650
8    1569.760799
9    6081.668239
dtype: float64
```

In [13]:
```
#easure the Variance
df.var()
```

```
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/3120059384.py:2: FutureWarning: Droppi
ng 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 ca
lling the reduction.
  df.var()
```

Out[13]:
```
Age                  5.978658e+01
ApplicantIncome      3.732039e+07
CoapplicantIncome    8.562930e+06
LoanAmount           7.325190e+03
Loan_Amount_Term     4.240668e+03
Credit_History       1.331362e-01
dtype: float64
```

In [14]:
```
#Measures the Interquartile Range (IQR)
from scipy.stats import iqr
iqr(df['Age'])
```

Out[14]:
```
nan
```

In [15]:
```
print(df.skew())
```

```
Age                 0.712146
ApplicantIncome     6.539513
CoapplicantIncome   7.491531
LoanAmount          2.677552
Loan_Amount_Term   -2.362414
Credit_History     -1.882361
dtype: float64
C:\Users\DELL\AppData\Local\Temp/ipykernel_7684/1926848427.py:1: FutureWarning: Droppi
ng 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 ca
lling the reduction.
  print(df.skew())
```

In [16]:
```
#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 betweer

#Approximately symmetric distribution: If the skewness value is between -½ and +½.
```

In [17]:
```
df.describe()
```

Out[17]:

	Age	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hist
count	513.000000	614.000000	614.000000	592.000000	600.00000	564.000
mean	32.101365	5403.459283	1621.245798	146.412162	342.00000	0.842
std	7.732178	6109.041673	2926.248369	85.587325	65.12041	0.364
min	24.000000	150.000000	0.000000	9.000000	12.00000	0.000
25%	25.000000	2877.500000	0.000000	100.000000	360.00000	1.000
50%	30.000000	3812.500000	1188.500000	128.000000	360.00000	1.000
75%	38.000000	5795.000000	2297.250000	168.000000	360.00000	1.000
max	56.000000	81000.000000	41667.000000	700.000000	480.00000	1.000

In [18]:
```
df.describe(include='all')
```

Out[18]:

	Loan_ID	Gender	Age	Married	Dependents	Education	Self_Employed	ApplicantInco
count	614	601	513.000000	611	599	614	582	614.000
unique	614	2	NaN	2	4	2	2	N
top	LP001002	Male	NaN	Yes	0	Graduate	No	N
freq	1	489	NaN	398	345	480	500	N
mean	NaN	NaN	32.101365	NaN	NaN	NaN	NaN	5403.459
std	NaN	NaN	7.732178	NaN	NaN	NaN	NaN	6109.041
min	NaN	NaN	24.000000	NaN	NaN	NaN	NaN	150.000
25%	NaN	NaN	25.000000	NaN	NaN	NaN	NaN	2877.500

	Loan_ID	Gender	Age	Married	Dependents	Education	Self_Employed	ApplicantInco
50%	NaN	NaN	30.000000	NaN	NaN	NaN	NaN	3812.500
75%	NaN	NaN	38.000000	NaN	NaN	NaN	NaN	5795.000
max	NaN	NaN	56.000000	NaN	NaN	NaN	NaN	81000.000

In [19]:
```
df.groupby('Age').count()
```

Out[19]:

Age	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplican
24.0	22	21	21	19	22	20	22	
25.0	107	105	107	105	107	102	107	
26.0	87	86	87	87	87	81	87	
27.0	23	23	22	22	23	22	23	
28.0	4	4	4	4	4	2	4	
30.0	53	53	53	52	53	52	53	
31.0	23	22	23	20	23	23	23	
32.0	18	18	18	18	18	18	18	
35.0	7	7	7	6	7	7	7	
37.0	18	17	18	18	18	16	18	
38.0	23	23	23	23	23	23	23	
40.0	22	20	22	22	22	20	22	
42.0	4	4	4	4	4	4	4	
43.0	45	44	45	42	45	40	45	
45.0	31	31	30	30	31	30	31	
46.0	6	6	6	6	6	5	6	
47.0	18	17	18	18	18	18	18	
50.0	1	1	1	1	1	1	1	
56.0	1	1	1	1	1	1	1	

In [20]:
```
df.groupby('Age')['ApplicantIncome']
```

Out[20]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x000001F7E7007CA0>
```

In [21]:
```
df.groupby('Age')['ApplicantIncome'].sum()
```

Out[21]:
```
Age
24.0    109185
25.0    538078
26.0    423267
27.0    178232
```

```
28.0    12637
30.0   314638
31.0   174155
32.0    73848
35.0    64332
37.0    99041
38.0   170038
40.0    87414
42.0    28751
43.0   205937
45.0   142359
46.0    38018
47.0   138701
50.0     4106
56.0     6540
Name: ApplicantIncome, dtype: int64
```

In [22]:

```
data = {'Gender':['m','f','f','m','f','m','m'],'Age':[24,25,26,27,28,30,32]}
df_sample = pd.DataFrame(data)
df_sample
```

Out[22]:

	Gender	Age
0	m	24
1	f	25
2	f	26
3	m	27
4	f	28
5	m	30
6	m	32

In [23]:

```
f_filter = df_sample['Gender']=='f'
print(df_sample[f_filter])

m_filter = df_sample['Gender']=='m'
print(df_sample[m_filter])
```

```
  Gender  Age
1      f   25
2      f   26
4      f   28
  Gender  Age
0      m   24
3      m   27
5      m   30
6      m   32
```

In [ ]:

In [ ]: