ASSIGNMENT 1

BASIC STATISTICS LEVEL 1

1) Libraries imported:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
sb.set()
```

Matplotlib.pyplot is the library we use to get the visualizations required

2) Loading the dataset & calling it:

```
a=pd.read_csv("/content/drive/MyDrive/Assignments/basic stats
1/sales_data_with_discounts.csv")
a.head()
```

	Date	Day	SKU	City	Volume	BU	Brand	Model	Avg Price	Total Sales Value	Discount Rate (%)	Discount Amount	Net Sales Value
0	01-04-2021	Thursday	M01	C	15	Mobiles	RealU	RU-10	12100	181500	11.654820	21153.498820	160346.501180
1	01-04-2021	Thursday	M02	C	10	Mobiles	RealU	RU-9 Plus	10100	101000	11.560498	11676.102961	89323.897039
2	01-04-2021	Thursday	M03	C	7	Mobiles	YouM	YM-99	16100	112700	9.456886	10657.910157	102042.089843
3	01-04-2021	Thursday	M04	C	6	Mobiles	YouM	YM-99 Plus	20100	120600	6.935385	8364.074702	112235.925298
4	01-04-2021	Thursday	M05	C	3	Mobiles	YouM	YM-98	8100	24300	17.995663	4372.946230	19927.053770

```
a.tail()
```

	Date	Day	SKU	City	Volume	BU	Brand	Model	Avg Price	Total Sales Value	Discount Rate (%)	Discount Amount	Net Sales Value
445	15-04-2021	Thursday	L06	C	2	Lifestyle	Jeera	M-Casuals	1300	2600	15.475687	402.367873	2197.632127
446	15-04-2021	Thursday	L07	C	6	Lifestyle	Viva	W-Western	2600	15600	17.057027	2660.896242	12939.103758
447	15-04-2021	Thursday	L08	C	2	Lifestyle	Viva	W-Lounge	1600	3200	18.965550	606.897606	2593.102394
448	15-04-2021	Thursday	L09	C	3	Lifestyle	Jeera	M-Formals	1900	5700	16.793014	957.201826	4742.798174
449	15-04-2021	Thursday	L10	C	1	Lifestyle	Jeera	M-Shoes	3100	3100	15.333300	475.332295	2624.667705

Here a is the dataframe and a.head() gives the first 5 entries of the df and a.tail() gives you the last 5 entries of the df

3) Numerical columns of the dataset:

```
b=a.select_dtypes(include=np.number)
b.head()
```

	Volume	Avg Price	Total Sales Value	Discount Rate (%)	Discount Amount	Net Sales Value
0	15	12100	181500	11.654820	21153.498820	160346.501180
1	10	10100	101000	11.560498	11676.102961	89323.897039
2	7	16100	112700	9.456886	10657.910157	102042.089843
3	6	20100	120600	6.935385	8364.074702	112235.925298
4	3	8100	24300	17.995663	4372.946230	19927.053770

4) Calculating Mean, Median and Mode for these numerical columns :

```
a.describe()
```

	Volume	Avg Price	Total Sales Value	Discount Rate (%)	Discount Amount	Net Sales Value
count	450.000000	450.000000	450.000000	450.000000	450.000000	450.000000
mean	5.066667	10453.433333	33812.835556	15.155242	3346.499424	30466.336131
std	4.231602	18079.904840	50535.074173	4.220602	4509.902963	46358.656624
min	1.000000	290.000000	400.000000	5.007822	69.177942	326.974801
25%	3.000000	465.000000	2700.000000	13.965063	460.459304	2202.208645
50%	4.000000	1450.000000	5700.000000	16.577766	988.933733	4677.788059
75%	6.000000	10100.000000	53200.000000	18.114718	5316.495427	47847.912852
max	31.000000	60100.000000	196400.000000	19.992407	25738.022194	179507.479049

a.describe() is the function we call in here to find out the statistical values for the numerical features in the given datasets that also include finding the count, mean,standard deviation , min value, max value, and the quartile values

but this doesn't give the mode so we need to find the mode separately

```
[10] a.Volume.mode()

0    3
Name: Volume, dtype: int64

a["Avg Price"].mode()

0     400
1     450
2     500
3    1300
4    8100
Name: Avg Price, dtype: int64

[12] a["Total Sales Value"].mode()

0    24300
Name: Total Sales Value, dtype: int64
```

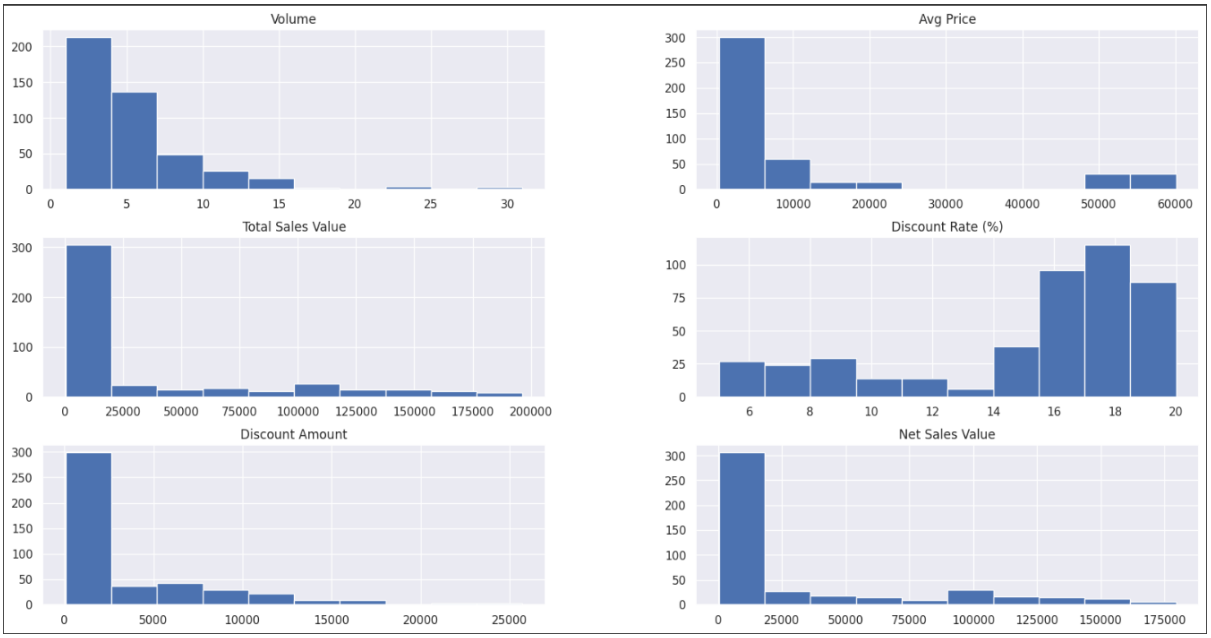
```
a["Discount Rate (%)"].mode().head(2)

0    5.007822
1    5.055218
Name: Discount Rate (%), dtype: float64

[14] a["Discount Amount"].mode().head(2)

0    69.177942
1    73.025199
Name: Discount Amount, dtype: float64

a["Net Sales Value"].mode().head(2)

0    326.974801
1    330.822058
Name: Net Sales Value, dtype: float64
```

MODE OF THE NUMERICAL FEATURE VARY FROM EACH OTHER DIFFERENTLY

- VOLUME IS 3
- AVG PRICE HAS -0 400, -1 450, -2 500, -3 1300, -4 8100,
- TOTAL SALES VALUE HAS 23400
- and the interesting fact is that the other three features such as discount rate, discount amount and net sales value, these three has unique value for every single entry so as there is no repetative value hence every value is considered as mode

5) Plotting histogram for all these numerical values:

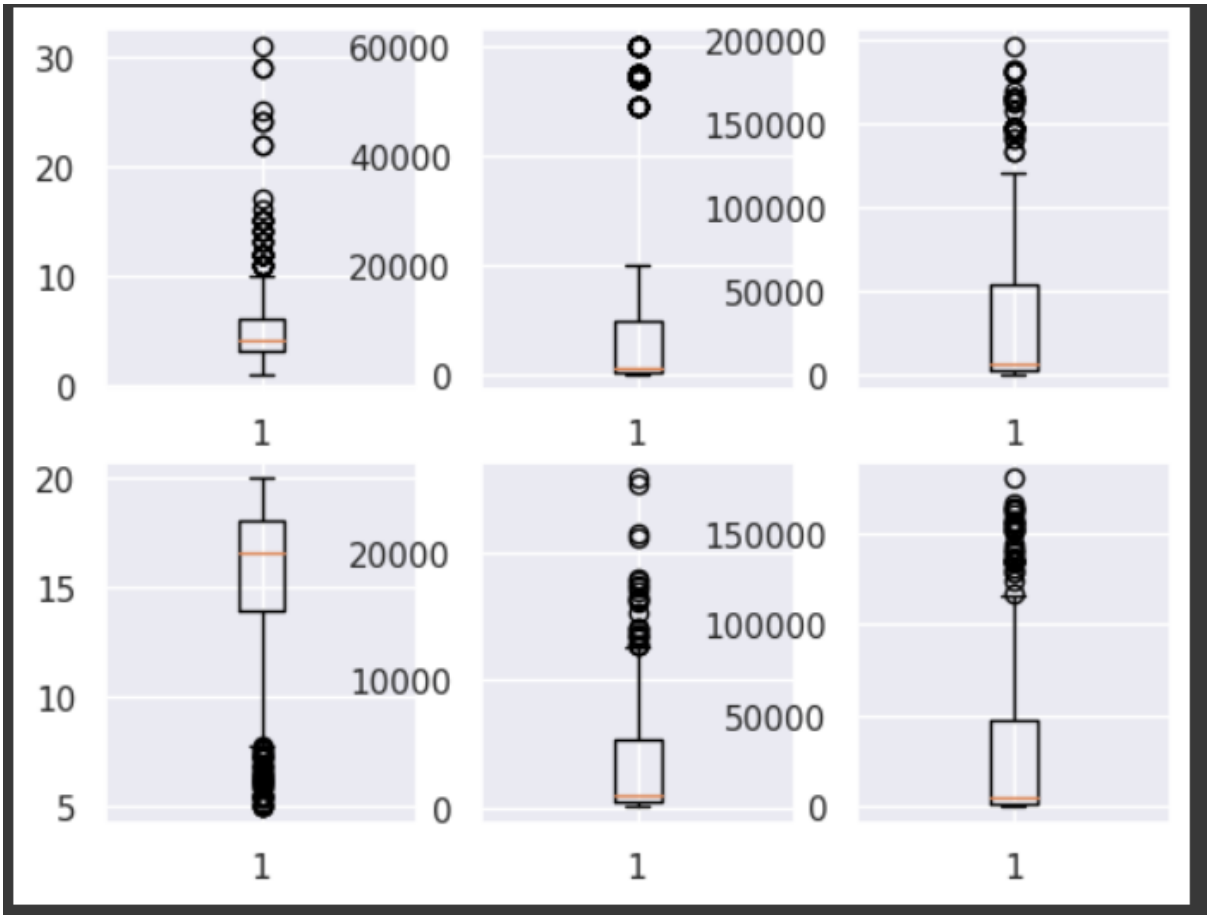
```
a.hist(figsize=(20,10))
plt.show()
```



6) Create boxplots for numerical variables to identify outliers and the interquartile range:

```
plt.subplot(2,3,1 )
plt.boxplot(a["Volume"])

plt.subplot(2,3,2)
plt.boxplot(a["Avg Price"])

plt.subplot(2,3,3)
plt.boxplot(a["Total Sales Value"])

plt.subplot(2,3,4)
plt.boxplot(a["Discount Rate (%)"])

plt.subplot(2,3,5)
plt.boxplot(a["Discount Amount"])

plt.subplot(2,3,6)
plt.boxplot(a["Net Sales Value"])

plt.show()
```

here using subplot method, plotted all the box plots for the numerical values at once



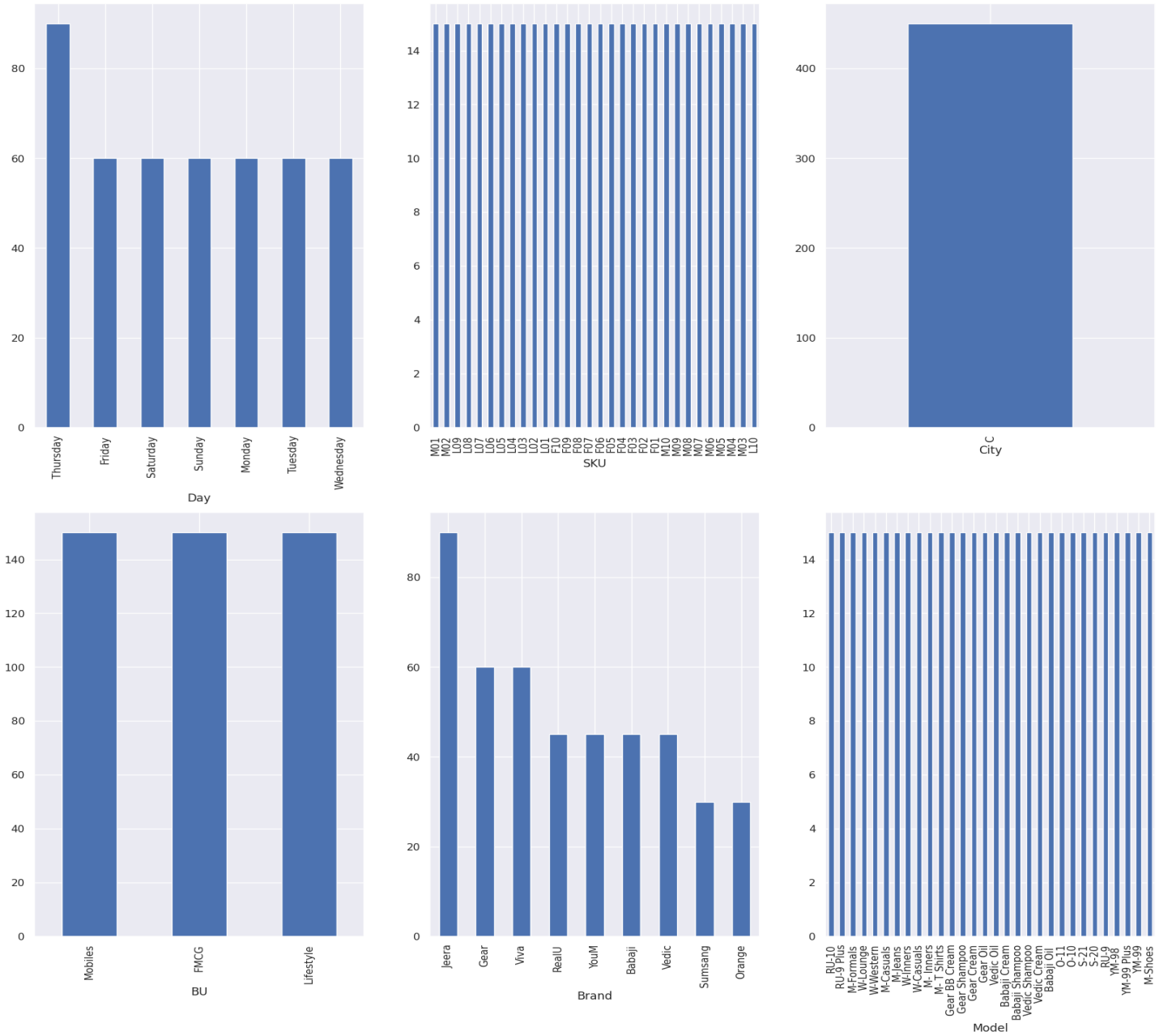
7) Finding out the categorical data in the given dataset:

```
a.select_dtypes(["object"])
```

	Date	Day	SKU	City	BU	Brand	Model
0	01-04-2021	Thursday	M01	C	Mobiles	RealU	RU-10
1	01-04-2021	Thursday	M02	C	Mobiles	RealU	RU-9 Plus
2	01-04-2021	Thursday	M03	C	Mobiles	YouM	YM-99
3	01-04-2021	Thursday	M04	C	Mobiles	YouM	YM-99 Plus
4	01-04-2021	Thursday	M05	C	Mobiles	YouM	YM-98

8) Plotting Bar-Plot for all the categorical data:

```
plt.figure(figsize=(20,20))
plt.subplot(2,3,1)
a['Day'].value_counts().plot(kind='bar')
plt.subplot(2,3,2)
a['SKU'].value_counts().plot(kind='bar')
plt.subplot(2,3,3)
a['City'].value_counts().plot(kind='bar')
plt.subplot(2,3,4)
a['BU'].value_counts().plot(kind='bar')
plt.subplot(2,3,5)
a['Brand'].value_counts().plot(kind='bar')
plt.subplot(2,3,6)
a['Model'].value_counts().plot(kind='bar')
```



9)

Standardization of Numerical Variables:

Standardization transforms/scaling data into a standard format, making it easier for computers to use and understand.

z-score normalisation:

Z score standardization uses mean and standard deviation from given data to standardize.

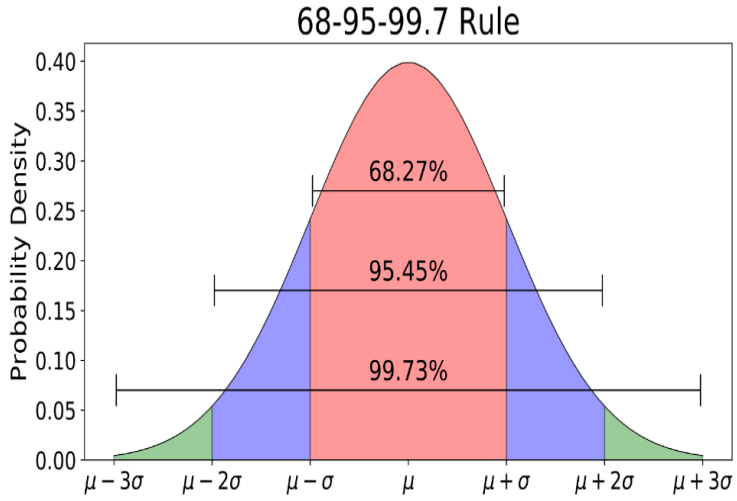
If z-score is 0 than data point is at mean.

If z-score is 1. Tells data point lies 1standard deviation more than mean

If z-score is -2. Tells data point lies 2standard deviation less than mean

$$Z = x - \mu / \sigma$$

X -data point, μ - mean , σ – Standard deviation




```
b=a.select_dtypes(include=np.number)
b.head()
```

	Volume	Avg Price	Total Sales Value	Discount Rate (%)	Discount Amount	Net Sales Value
0	15	12100	181500	11.654820	21153.498820	160346.501180
1	10	10100	101000	11.560498	11676.102961	89323.897039
2	7	16100	112700	9.456886	10657.910157	102042.089843
3	6	20100	120600	6.935385	8364.074702	112235.925298
4	3	8100	24300	17.995663	4372.946230	19927.053770

so we saved the numerical values into b so the we can check the difference between the original and standardized values

the above values are the ones before standardization

```
#STANDARDIZATION

def get_stand(x):
  return ((x-x.mean())/(x.std()))
```

here we defined a function to standardize the values and it is the implementation of the formula that is discussed above

```
get_stand(b).head()
```

	Volume	Avg Price	Total Sales Value	Discount Rate (%)	Discount Amount	Net Sales Value
0	2.347417	0.091072	2.922469	-0.829365	3.948422	2.801638
1	1.165831	-0.019548	1.329516	-0.851714	1.846958	1.269613
2	0.456880	0.312312	1.561038	-1.350129	1.621190	1.543957
3	0.220563	0.533552	1.717365	-1.947555	1.112568	1.763847
4	-0.488389	-0.130168	-0.188242	0.672990	0.227598	-0.227342

These are the values after standardization, and to remember we use standardization to make the data scale free

10) Creating Dummy Variables:

Dummy variables are used to convert the categorical data into numerical data, or we can also say the dummy variables are the numerical representation of the categorical data

We use dummy variables so we can apply the machine learning algorithms to the converted numerical data

Let’s say the brand feature has certain different inputs in it but none of them are in numerical form and we cannot apply any function to it

So while converting them to dummy variables, we have different types of method and we precisely use “ONE HOT ENCODING” because we are making dummy variables in the input features side and “OHE” is mostly used for the input features transformation

While doing this encoding, the active entry will represented as 1 and the remaining or the inactive ones will be kept 0

```
#creating dummy variables
pd.get_dummies(a["Brand"],dtype="int")
```

	Babaji	Gear	Jeera	Orange	RealU	Sumsang	Vedic	Viva	YouM
0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...
445	0	0	1	0	0	0	0	0	0
446	0	0	0	0	0	0	0	1	0
447	0	0	0	0	0	0	0	1	0
448	0	0	1	0	0	0	0	0	0
449	0	0	1	0	0	0	0	0	0

450 rows × 9 columns

So this is how it looks when we create the dummy variables for the categorical data