1. Mean, Median & Mode (MMM) Table Inference

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
Mean	108	67.3034	66.3332	66.3702	72.1006	62.2782	288655
Median	108	67	65	66	71	62	265000
Mode	1	62	63	65	60	56.7	300000

Inference Tabulation:

Index	Mean	Median	Outlier Status	Mode
Ssc_p	The average ssc pass mark of 215 students = 67.3034	The average ssc pass mark of 215 students excluding outliers = 67	Not present	Most repeated pass mark secured in ssc among 215 students= 62
Take away	**Average performance in Ssc**		-	-
Hsc_p	The average hsc pass mark of 215 students = 66.3332	The average hsc pass mark of 215 students excluding outliers = 65	Present (Lesser & Greater)	Most repeated pass mark secured in hsc among 215 students= 63
Take away	**Average performance in Hsc**		-	-
Degree_p	The average degree pass mark of 215 students= 66.3702	The average degree pass mark of 215 students excluding outliers = 66	Present (Greater)	Most repeated pass mark secured in degree among 215 students= 65
Take away	**Average performance in degree graduation**		-	-
Etest_p	The average e-test pass mark of 215 students = 72.1006	The average e-test pass mark of 215 students excluding outliers = 71	Not Present	Most repeated pass mark secured in e-test among 215 students= 60
Take away	**Above average performance in Etest**		-	-
Mba_p	The average MBA pass mark of 215 students = 62.2782	The average MBA pass mark of 215 students excluding outliers = 62	Not present	Most repeated pass mark secured in MBA among 215 students= 56.7
Take away	**Average performance in MBA**		-	-
Salary	The average salary of 215 students = RS.2,88,655	The average salary of 215 students excluding outliers= RS.2,65,000	Present (Greater)	Most repeated salary package received among 215 students= RS.3,00,000
Take away	**Among 215 students, most of the students receive RS.3,00,000 as their salary.**			

2. Formula Reasoning:

IQR = Q3-Q1

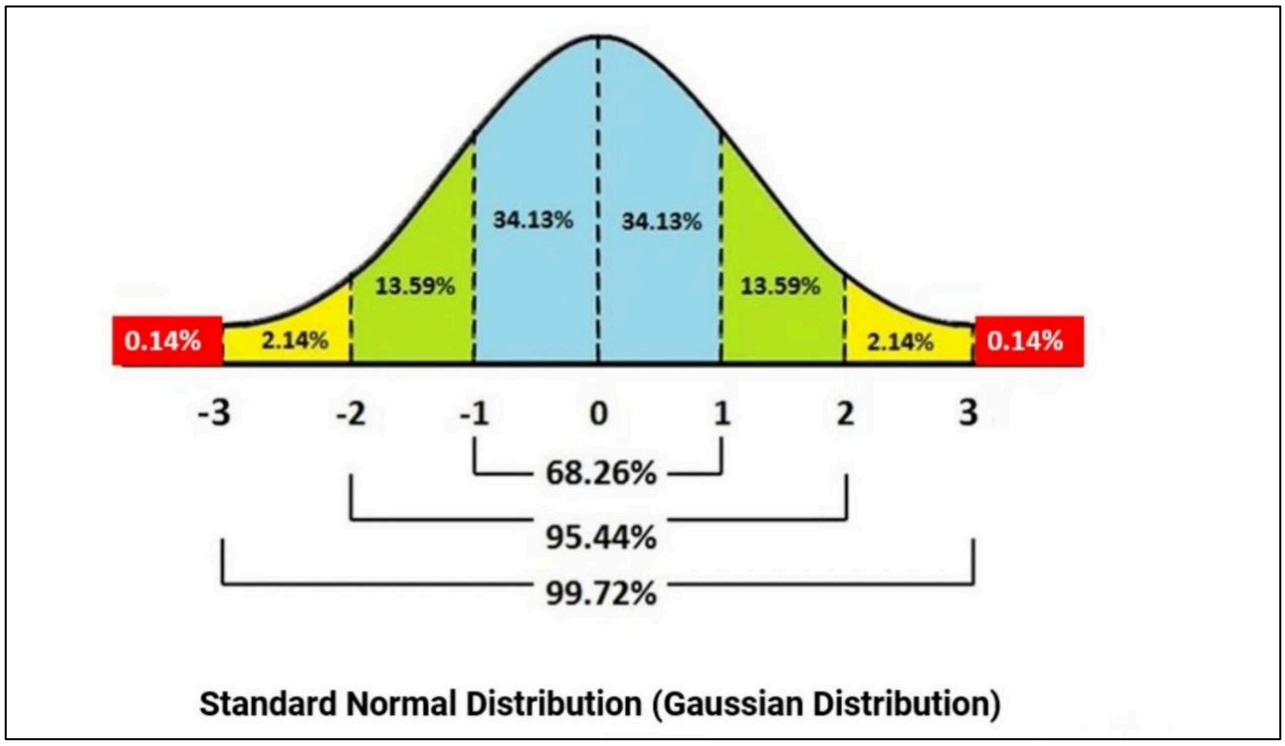
Lesser Outlier
Less Than
Outlier range= Q1- 1.5*IQR

Greater Outlier
Greater Than
Outlier range= Q3+ 1.5*IQR

Question : Why 1.5 scale is multiplied with IQR for outliers?

Reason:

Consider the Gaussian distribution decision range for outlier detection as shown below.



Standard Normal Distribution (Gaussian Distribution)

Here we are looking for the data which is useful for us which lies within the range mean plus or minus 3 sigma data points. Those data points which lies beyond this range is considered as outlier.

To achieve this let's try to calculate the approximate scale value which has to be multiplied with IQR so that we end up with ±3σ range.

Consider Scale = 1

Lower Bound = Q1 - 1 * IQR

= Q1 - 1 * (Q3 - Q1)

= -0.675σ - 1 * (0.675 - [-0.675]) σ

= -0.675σ - 1 * 1.35σ

= **-2.025σ**

Upper Bound = Q3 + 1 * IQR

= Q3 + 1 * (Q3 - Q1)

= 0.675σ + 1 * (0.675 - [-0.675])σ

= 0.675σ + 1 * 1.35σ

= **2.025σ**

Conclusion:

If the scale is taken as 1, then according to IQR Method any data which lies beyond 2.025σ from the mean (μ), on either side, shall be considered an outlier. But as we know, up to 3σ, on either side of the μ, the data is useful. So, we cannot take scale = 1, because the decision range gets so small (compared to 3σ) that it considers some data points as outliers, which is not desirable

Consider Scale = 1.5

Lower Bound = Q1 - 1.5 * IQR

= Q1 - 1.5 * (Q3 - Q1)

= -0.675σ - 1.5* (0.675 - [-0.675]) σ

= -0.675σ - 1.5 * 1.35σ

= **-2.7σ**

Upper Bound = Q3 + 1.5 * IQR

= Q3 + 1.5 * (Q3 - Q1)

= 0.675σ + 1.5 * (0.675 - [-0.675])σ

= 0.675σ + 1.5 * 1.35σ

= **2.7σ**

Conclusion:

If the scale is taken as 1.5, then according to IQR Method any data which lies beyond 2.7σ from the mean (μ), on either side, shall be considered an outlier. But as we know, up to 3σ, on either side of the μ, the data is useful. So, if we take scale = 1.5, this makes the decision range closest to what Gaussian Distribution considers for outlier detection, and this is exactly what we wanted. So, it is desirable.

Consider Scale = 1.7

Lower Bound = Q1 - 1.7 * IQR

= Q1 - 1.7 * (Q3 - Q1)

= -0.675σ - 1.7* (0.675 - [-0.675]) σ

= -0.675σ - 1.7 * 1.35σ

= **-2.97σ**

Upper Bound = Q3 + 1.7 * IQR

= Q3 + 1.7 * (Q3 - Q1)

= 0.675σ + 1.7 * (0.675 - [-0.675])σ

= 0.675σ + 1.7 * 1.35σ

= **2.97σ**

Conclusion:

If the scale is taken as 1.7, then according to IQR Method any data which lies beyond 2.97σ from the mean (μ), on either side, shall be considered an outlier. But as we know, up to 3σ, on either side of the μ, the data is useful. So, if we take scale = 1.7, this makes the decision range the same as that of Gaussian Distribution considers for outlier detection. So, it is desirable again. But IQR is not the perfect method to detect outliers.

To keep the tolerance we choose 1.5 scale on safer side though 1.7 scale suits the best.

Consider Scale = 2

Lower Bound = Q1 - 2 * IQR

= Q1 - 2 * (Q3 - Q1)

= -0.675σ - 2 * (0.675 - [-0.675]) σ

= -0.675σ - 2 * 1.35σ

= **-3.375σ**

Upper Bound = Q3 + 2 * IQR

= Q3 + 2 * (Q3 - Q1)

= 0.675σ + 2 * (0.675 - [-0.675])σ

= 0.675σ + 2 * 1.35σ

= **3.375σ**

Conclusion:

If the scale is taken as 2, then according to IQR Method any data which lies beyond 3.375σ from the mean (μ), on either side, shall be considered an outlier. But as we know, up to 3σ, on either side of the μ, the data is useful. So, we cannot take scale = 2, because the decision range gets so big (compared to 3σ) that it considers some outliers as data points, which is not desirable again.

3. Problem on IQR

Question :

a. The interquartile range. Compare the two interquartile ranges.

b. Any outliers in either set.

The five number summary for the day and night classes is

	Minimum	Q_1	Median	Q_3	Maximum
Day	32	56	74.5	82.5	99
Night	25.5	78	81	89	98

Solution :

Classes	Q_1	Q_3	IQR= Q_3- Q_1	Lesser outlier range = Q_1-1.5*IQR	Greater outlier range = Q_3 + 1.5*IQR	Upper & Lower Range
Day	56	82.5	26.5	16.25	122.25	**Range:(16.25, 122.25)** No values in the row named day is outside the range. So no outliers are present.
Night	78	89	11	61.5	105.5	**Range:(61.5, 105.5)** Value 25.5 in the row named night is outside the range. So outlier is present. **Type of outlier : Low range outlier**

4.Kurtosis & Skewness Table Inference

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
kurtosis	-1.2	-0.60751	0.0869008	-0.0974897	-1.08858	-0.470723	-0.239837
skew	0	-0.132649	0.162611	0.204164	0.282308	0.313576	0.8067

Inference:

Kurtosis: (Tells about the Tailness)

Kurtosis value for all the columns starting from ssc_p to salary is <3 which means the type of kurtosis is platykurtic(Negative Excess Kurtosis).

Which in turn gives the information that the number of students falling under below average & above average (to some extent) whereas number of students falling under average (to better extent) in terms of exam performance & receiving salary, where the most population of students fall under average criteria.

Skewness: (Tells us about Asymmetry from the normal distribution)

Ssc_p alone exhibits negative skewness.i.e, Above average scorers are more in count when compared to below average scorers.

Hsc_p, Degree_p, Etest_p & MBA _p variables exhibits positive skewness. i.e Above average scorers are lesser in count when compared to below average scorers.

Salary variable also exhibits positive skewness. i.e Above average salary receivers are lesser in count when compared to below average salary receivers.

5. Probability Distribution Function (PDF) Code Explanation:

jupyter PDF Code Explanation Last Checkpoint: 7 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown

In [1]:

```
#Generating a function for PDF such that getting the aruguments from the user
def get_pdf_probability(dataset,startrange,endrange):

    #Importing necessary libraries for visual representation
    from matplotlib import pyplot
    from scipy.stats import norm
    import seaborn as sns


    # Plotting distribution plot with density curve in blue color & histogram in Green colour
    ax = sns.distplot(dataset,kde=True,kde_kws={'color':'blue'},color='Green')


    # Making initial & end range axis vertical line visible
    pyplot.axvline(startrange,color='Red')
    pyplot.axvline(endrange,color='Red')


    # Generating a sample
    sample = dataset


    # Calculating parameters like mean & std and displaying the calulated values using print statement
    sample_mean =sample.mean()
    sample_std = sample.std()
    print('Mean=%.3f, Standard Deviation=%.3f' % (sample_mean, sample_std))


    # Defining the distribution by giving mean & std as input
    dist = norm(sample_mean, sample_std)

    # Sample probabilities for a range of outcomes
    values = [value for value in range(startrange, endrange)]
    probabilities = [dist.pdf(value) for value in values]

    # For whole probality
    prob=sum(probabilities)
    print("The area between range({},{}):{}".format(startrange,endrange,sum(probabilities)))
    return prob
```

Graph

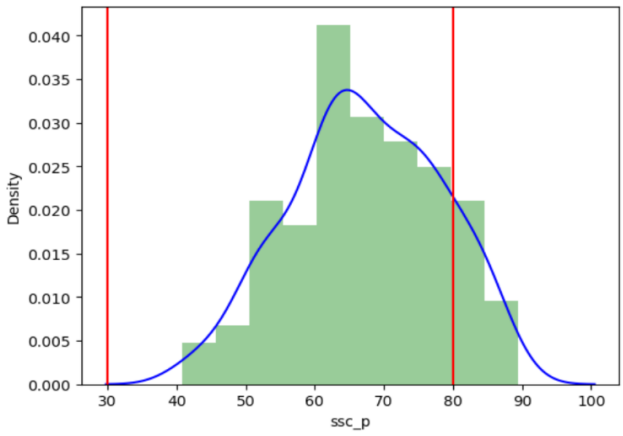
In [8]:

```
get_pdf_probability(dataset["ssc_p"],30,80)
```

```
Mean=67.303, Standard Deviation=10.827
The area between range(30,80):0.8698639973987944
```

```
/var/folders/07/ykgp85052b11h5kz22ghn8l40000gn/T/ipykernel_43901/2842244316.py:5: Us

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(dataset,kde=True,kde_kws={'color':'blue'},color='Green')
```

Out[8]: 0.8698639973987944

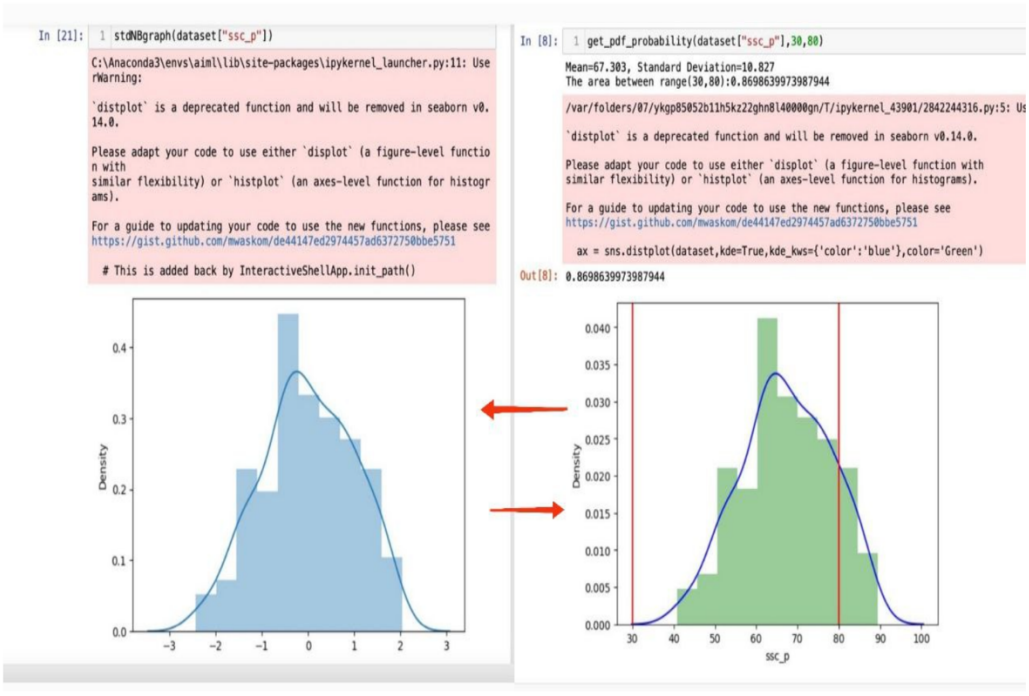


முத்து வசுமதி

6. Standard Normal Distribution (SND) Code Explanation:

```
In [1]:
# Creating function for SND graph
def stdNBgraph(dataset):


    # Converted to standard Normal Distribution
    #Importing necessary library
    import seaborn as sns

    # Calculating parameters like mean & std the required coloumns
    mean=dataset.mean()
    std=dataset.std()


    #Converting the  required coloumn of the dataset into list
    values=[i for i in dataset]


    #Calculating Z Score for the Values obtained in the previous step and storing in the list
    z_score=[((j-mean)/std) for j in values]


    #Feeding the Z score value into the distance plot for creating visual representation
    sns.distplot(z_score,kde=True)

    x = sum(z_score)/len(z_score)
    print(x)
    #z_score.std()
```

Graph Comparison (SND & PDF)



7. For Loop in Traditional Way

One Liner For Loop:

```
# sample probabilities for a range of outcomes
values = [value for value in range(startrange, endrange)]
probabilities = [dist.pdf(value) for value in values]
```

Tradition Way For Loop:

```
#values = [value for value in range(startrange, endrange)]

values=[]
for value in range(startrange, endrange):
    values.append(value)

#probabilities = [dist.pdf(value) for value in values]

probabilities = []
for value in values:
    probabilities.append(dist.pdf(value))
```

Proof:

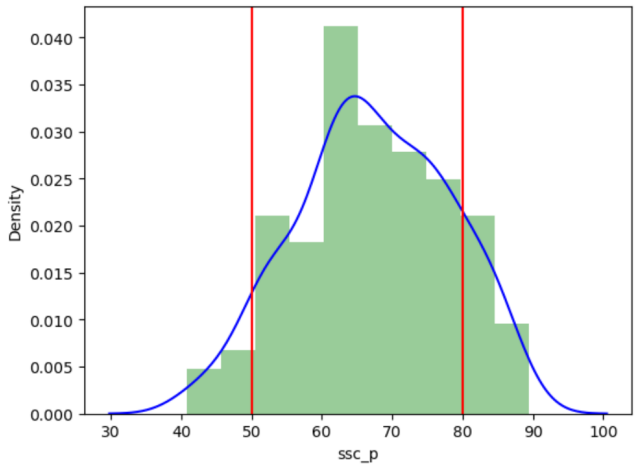
In [13]:
```
def get_pdf_probability(dataset,startrange, endrange):
    from matplotlib import pyplot
    from scipy.stats import norm
    import seaborn as sns
    ax = sns.distplot(dataset,kde=True,kde_kws={'color':'blue'},color='Green')
    pyplot.axvline(startrange,color='Red')
    pyplot.axvline(endrange,color='Red')
    # generate a sample
    sample = dataset
    # calculate parameters
    sample_mean =sample.mean()
    sample_std = sample.std()
    print('Mean=%.3f, Standard Deviation=%.3f' % (sample_mean, sample_std))
    # define the distribution
    dist = norm(sample_mean, sample_std)

    # sample probabilities for a range of outcomes

    #values = [value for value in range(startrange, endrange)]

    values=[]
    for value in range(startrange, endrange):
        values.append(value)

    #probabilities = [dist.pdf(value) for value in values]

    probabilities = []
    for value in values:
        probabilities.append(dist.pdf(value))

    prob=sum(probabilities)
    print("The area between range({},{}):{}".format(startrange,endrange,sum(probabilities)))
    return prob

```

In [14]:
```
get_pdf_probability(dataset["ssc_p"],50,80)
```

```
Mean=67.303, Standard Deviation=10.827
The area between range(50,80):0.8201083468562449
```

```
/var/folders/07/ykgp85052b11h5kz22ghn8l40000gn/T/ipykernel_98036/2713336809.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(dataset,kde=True,kde_kws={'color':'blue'},color='Green')
```

Out[14]:
```
0.8201083468562449
```



8.CKD File Preprocessing

jupyter Preprocessed Kidney Disease File Last Checkpoint: 6 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
dataset=pd.read_csv("kidney_disease.csv")
dataset
```

Out[2]:

	id	age	F	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44.0	7800.0	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38.0	6000.0	NaN	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31.0	7500.0	NaN	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32.0	6700.0	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35.0	7300.0	4.6	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	47.0	6700.0	4.9	no	no	no	good	no	no	notckd
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	54.0	7800.0	6.2	no	no	no	good	no	no	notckd
397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	49.0	6600.0	5.4	no	no	no	good	no	no	notckd
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	51.0	7200.0	5.9	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	53.0	6800.0	6.1	no	no	no	good	no	no	notckd

400 rows × 26 columns

In [3]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              400 non-null    int64
 1   age             391 non-null    float64
 2   F               388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object
 7   pc              335 non-null    object
 8   pcc             396 non-null    object
 9   ba              396 non-null    object
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             329 non-null    float64
 17  wc              294 non-null    float64
 18  rc              269 non-null    float64
 19  htn             398 non-null    object
 20  dm              398 non-null    object
 21  cad             398 non-null    object
 22  appet           399 non-null    object
 23  pe              399 non-null    object
 24  ane             399 non-null    object
 25  classification  400 non-null    object
dtypes: float64(14), int64(1), object(11)
memory usage: 81.4+ KB
```

In [4]:
```
dataset.isnull().sum().T
#print(a,end='')
```

Out[4]:
```
id                0
age               9
F                12
sg               47
al               46
su               49
rbc             152
pc               65
pcc               4
ba                4
bgr              44
bu               19
sc               17
sod              87
pot              88
hemo             52
pcv              71
wc              106
rc              131
htn               2
dm                2
cad               2
appet             1
pe                1
ane               1
classification    0
dtype: int64
```

Deleting Certain Rows with Missing Values

In [5]:
```
dataset.dropna(subset=['age','pcc', 'ba','htn','dm','cad','appet','pe','ane'], inplace=True)
dataset
```

Out[5]:

	id	age	F	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44.0	7800.0	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38.0	6000.0	NaN	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31.0	7500.0	NaN	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32.0	6700.0	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35.0	7300.0	4.6	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	47.0	6700.0	4.9	no	no	no	good	no	no	notckd
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	54.0	7800.0	6.2	no	no	no	good	no	no	notckd
397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	49.0	6600.0	5.4	no	no	no	good	no	no	notckd
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	51.0	7200.0	5.9	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	53.0	6800.0	6.1	no	no	no	good	no	no	notckd

384 rows × 26 columns

Filling Missing Numerical Values with Median Values

In [6]:
```
dataset.fillna(dataset.median(), inplace=True)
dataset
```

```
/var/folders/07/ykgp85052b11h5kz22ghn8l40000gn/T/ipykernel_3973/3113556935.py:1: FutureWarning: The default v
of numeric_only in DataFrame.median is deprecated. In a future version, it will default to False. In additior
cifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to
ce this warning.
  dataset.fillna(dataset.median(), inplace=True)
```

Out[6]:

	id	age	F	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44.0	7800.0	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38.0	6000.0	4.7	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31.0	7500.0	4.7	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32.0	6700.0	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35.0	7300.0	4.6	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	47.0	6700.0	4.9	no	no	no	good	no	no	notckd
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	54.0	7800.0	6.2	no	no	no	good	no	no	notckd
397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	49.0	6600.0	5.4	no	no	no	good	no	no	notckd
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	51.0	7200.0	5.9	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	53.0	6800.0	6.1	no	no	no	good	no	no	notckd

384 rows × 26 columns

In [7]:
```
dataset.isna().sum()
```

Out[7]:
```
id                   0
age                  0
F                    0
sg                   0
al                   0
su                   0
rbc                146
pc                  60
pcc                  0
ba                   0
bgr                  0
bu                   0
sc                   0
sod                  0
pot                  0
hemo                 0
pcv                  0
wc                   0
rc                   0
htn                  0
dm                   0
cad                  0
appet                0
pe                   0
ane                  0
classification       0
dtype: int64
```

Finding Mode to Replace Missing Categorical Values

In [8]:
```
categorical_columns = ['rbc','pc']
modes = dataset[categorical_columns].mode()
modes
```

Out[8]:

	rbc	pc
0	normal	normal

Replacing the Missing Categorical Values With Calculated Mode

In [9]:
```
dataset[dataset.select_dtypes(include=['object']).columns] = dataset.select_dtypes(include=['object']).ap
dataset
```

Out[9]:

	id	age	F	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	normal	normal	notpresent	notpresent	...	44.0	7800.0	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	normal	normal	notpresent	notpresent	...	38.0	6000.0	4.7	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31.0	7500.0	4.7	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32.0	6700.0	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35.0	7300.0	4.6	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	47.0	6700.0	4.9	no	no	no	good	no	no	notckd
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	54.0	7800.0	6.2	no	no	no	good	no	no	notckd
397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	49.0	6600.0	5.4	no	no	no	good	no	no	notckd
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	51.0	7200.0	5.9	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	53.0	6800.0	6.1	no	no	no	good	no	no	notckd

384 rows × 26 columns

In [10]:
```
dataset.isnull().sum()
```

Out[10]:
```
id                0
age               0
F                 0
sg                0
al                0
su                0
rbc               0
pc                0
pcc               0
ba                0
bgr               0
bu                0
sc                0
sod               0
pot               0
hemo              0
pcv               0
wc                0
rc                0
htn               0
dm                0
cad               0
appet             0
pe                0
ane               0
classification    0
dtype: int64
```

Defining Quan & Qual for Outlier Removal

In [11]:
```
def quanQual(dataset):
    quan=[]
    qual=[]
    for columnName in dataset.columns:
        #print(columnName)
        if(dataset[columnName].dtype=='O'):
            #print("qual")
            qual.append(columnName)
        else:
            #print("quan")
            quan.append(columnName)
    return quan,qual
```

In [12]:
```
quan,qual=quanQual(dataset)
quan
```

Out[12]:
```
['id',
 'age',
 'F',
 'sg',
 'al',
 'su',
 'bgr',
 'bu',
 'sc',
 'sod',
 'pot',
 'hemo',
 'pcv',
 'wc',
 'rc']
```

In [13]:
```
dataset[qual]
```

Out[13]:

	rbc	pc	pcc	ba	htn	dm	cad	appet	pe	ane	classification
0	normal	normal	notpresent	notpresent	yes	yes	no	good	no	no	ckd
1	normal	normal	notpresent	notpresent	no	no	no	good	no	no	ckd
2	normal	normal	notpresent	notpresent	no	yes	no	poor	no	yes	ckd
3	normal	abnormal	present	notpresent	yes	no	no	poor	yes	yes	ckd
4	normal	normal	notpresent	notpresent	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...
395	normal	normal	notpresent	notpresent	no	no	no	good	no	no	notckd
396	normal	normal	notpresent	notpresent	no	no	no	good	no	no	notckd
397	normal	normal	notpresent	notpresent	no	no	no	good	no	no	notckd
398	normal	normal	notpresent	notpresent	no	no	no	good	no	no	notckd
399	normal	normal	notpresent	notpresent	no	no	no	good	no	no	notckd

384 rows × 11 columns

Outlier Removal

In [14]:
```
descriptive=pd.DataFrame(index=["Mean","Median","Mode","Q1:25%","Q2:50%",
                                "Q3:75%","99%","Q4:100%","IQR","1.5rule","Lesser","Greater","Min","Max"],
for columnName in quan:
    descriptive[columnName]["Mean"]=dataset[columnName].mean()
    descriptive[columnName]["Median"]=dataset[columnName].median()
    descriptive[columnName]["Mode"]=dataset[columnName].mode()[0]
    descriptive[columnName]["Q1:25%"]=dataset.describe()[columnName]["25%"]
    descriptive[columnName]["Q2:50%"]=dataset.describe()[columnName]["50%"]
    descriptive[columnName]["Q3:75%"]=dataset.describe()[columnName]["75%"]
    descriptive[columnName]["99%"]=np.percentile(dataset[columnName],99)
    descriptive[columnName]["Q4:100%"]=dataset.describe()[columnName]["max"]
    descriptive[columnName]["IQR"]=descriptive[columnName]["Q3:75%"]-descriptive[columnName]["Q1:25%"]
    descriptive[columnName]["1.5rule"]=1.5*descriptive[columnName]["IQR"]
    descriptive[columnName]["Lesser"]=descriptive[columnName]["Q1:25%"]-descriptive[columnName]["1.5rule
    descriptive[columnName]["Greater"]=descriptive[columnName]["Q3:75%"]+descriptive[columnName]["1.5rul
    descriptive[columnName]["Min"]=dataset[columnName].min()
    descriptive[columnName]["Max"]=dataset[columnName].max()
```

In [15]:
```
lesser=[]
greater=[]

for columnName in quan:
    if(descriptive[columnName]["Lesser"]>descriptive[columnName]["Min"]):
        lesser.append(columnName)
    if(descriptive[columnName]["Greater"]<descriptive[columnName]["Q4:100%"]):
        greater.append(columnName)

```

In [16]:
```
lesser
```

Out[16]: ['age', 'F', 'sg', 'bgr', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc']

In [17]:
```
greater
```

Out[17]: ['F', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot', 'wc', 'rc']

In [18]:
```
for column in lesser:
    dataset[column][dataset[column]<descriptive[column]["Lesser"]]=descriptive[column]["Lesser"]
for column in greater:
    dataset[column][dataset[column]>descriptive[column]["Greater"]]=descriptive[column]["Greater"]

```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
/var/folders/07/ykgp85052b11h5kz22ghn8l40000gn/T/ipykernel_3973/3400726572.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#r
ing-a-view-versus-a-copy
  dataset[column][dataset[column]<descriptive[column]["Lesser"]]=descriptive[column]["Lesser"]
/var/folders/07/ykgp85052b11h5kz22ghn8l40000gn/T/ipykernel_3973/3400726572.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#r
```

In [19]:
```
descriptive=pd.DataFrame(index=["Mean","Median","Mode","Q1:25%","Q2:50%",
                                "Q3:75%","99%","Q4:100%","IQR","1.5rule","Lesser","Greater","Min","Max"],c
for columnName in quan:
    descriptive[columnName]["Mean"]=dataset[columnName].mean()
    descriptive[columnName]["Median"]=dataset[columnName].median()
    descriptive[columnName]["Mode"]=dataset[columnName].mode()[0]
    descriptive[columnName]["Q1:25%"]=dataset.describe()[columnName]["25%"]
    descriptive[columnName]["Q2:50%"]=dataset.describe()[columnName]["50%"]
    descriptive[columnName]["Q3:75%"]=dataset.describe()[columnName]["75%"]
    descriptive[columnName]["99%"]=np.percentile(dataset[columnName],99)
    descriptive[columnName]["Q4:100%"]=dataset.describe()[columnName]["max"]
    descriptive[columnName]["IQR"]=descriptive[columnName]["Q3:75%"]-descriptive[columnName]["Q1:25%"]
    descriptive[columnName]["1.5rule"]=1.5*descriptive[columnName]["IQR"]
    descriptive[columnName]["Lesser"]=descriptive[columnName]["Q1:25%"]-descriptive[columnName]["1.5rule"
    descriptive[columnName]["Greater"]=descriptive[columnName]["Q3:75%"]+descriptive[columnName]["1.5rule
    descriptive[columnName]["Min"]=dataset[columnName].min()
    descriptive[columnName]["Max"]=dataset[columnName].max()
```

In [20]:
```
descriptive
```

Out[20]:

	id	age	F	sg	al	su	bgr	bu	sc	sod	pot	hemo	pcv	
Mean	198.979167	51.574219	76.002604	1.017728	0.90625	0.0	134.552083	49.758984	2.061328	138.205729	4.378385	12.544434	39.114583	8142.
Median	198.5	55.0	80.0	1.02	0.0	0.0	122.0	42.0	1.3	138.0	4.4	12.6	40.0	
Mode	0	60.0	80.0	1.02	0.0	0.0	224.25	110.125	5.4	138.0	4.4	12.6	40.0	
Q1:25%	97.75	42.0	70.0	1.015	0.0	0.0	101.75	27.0	0.9	135.0	4.0	10.9	34.0	
Q2:50%	198.5	55.0	80.0	1.02	0.0	0.0	122.0	42.0	1.3	138.0	4.4	12.6	40.0	
Q3:75%	301.25	65.0	80.0	1.02	2.0	0.0	150.75	60.25	2.7	141.0	4.8	14.525	44.0	
99%	395.17	80.17	95.0	1.025	4.0	0.0	224.25	110.125	5.4	150.0	6.0	17.617	53.17	
Q4:100%	399.0	90.0	95.0	1.025	5.0	0.0	224.25	110.125	5.4	150.0	6.0	17.8	54.0	
IQR	203.5	23.0	10.0	0.005	2.0	0.0	49.0	33.25	1.8	6.0	0.8	3.625	10.0	
1.5rule	305.25	34.5	15.0	0.0075	3.0	0.0	73.5	49.875	2.7	9.0	1.2	5.4375	15.0	
Lesser	-207.5	7.5	55.0	1.0075	-3.0	0.0	28.25	-22.875	-1.8	126.0	2.8	5.4625	19.0	
Greater	606.5	99.5	95.0	1.0275	5.0	0.0	224.25	110.125	5.4	150.0	6.0	19.9625	59.0	

```
greater=[]

for columnName in quan:
    if(descriptive[columnName]["Lesser"]>descriptive[columnName]["Min"]):
        lesser.append(columnName)
    if(descriptive[columnName]["Greater"]<descriptive[columnName]["Q4:100%"]):
        greater.append(columnName)
```

In [22]:
```
lesser
```

Out[22]: []

In [23]:
```
greater
```

Out[23]: []

In [24]:
```
dataset
```

Out[24]:

	id	age	F	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.0200	1.0	0.0	normal	normal	notpresent	notpresent	...	44.0	7800.0	5.2000	yes	yes	no	good	no	no	ckd
1	1	7.5	55.0	1.0200	4.0	0.0	normal	normal	notpresent	notpresent	...	38.0	6000.0	4.7000	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.0100	2.0	0.0	normal	normal	notpresent	notpresent	...	31.0	7500.0	4.7000	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.0075	4.0	0.0	normal	abnormal	present	notpresent	...	32.0	6700.0	3.9000	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.0100	2.0	0.0	normal	normal	notpresent	notpresent	...	35.0	7300.0	4.6000	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	395	55.0	80.0	1.0200	0.0	0.0	normal	normal	notpresent	notpresent	...	47.0	6700.0	4.9000	no	no	no	good	no	no	notckd
396	396	42.0	70.0	1.0250	0.0	0.0	normal	normal	notpresent	notpresent	...	54.0	7800.0	5.8125	no	no	no	good	no	no	notckd
397	397	12.0	80.0	1.0200	0.0	0.0	normal	normal	notpresent	notpresent	...	49.0	6600.0	5.4000	no	no	no	good	no	no	notckd
398	398	17.0	60.0	1.0250	0.0	0.0	normal	normal	notpresent	notpresent	...	51.0	7200.0	5.8125	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.0250	0.0	0.0	normal	normal	notpresent	notpresent	...	53.0	6800.0	5.8125	no	no	no	good	no	no	notckd

384 rows × 26 columns

Saving the Preprocessed dataset as a New File

In [25]:
```
dataset.to_csv("PreProcessed_kidney_disease.csv",index=False)
```

---- The End----

முத்து வசுமதி