Importing libraries

from scipy.stats import norm, t  
import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
pd.options.display.max\_columns = 500  
pd.options.display.max\_rows = 500  
from statistics import stdev, mean  
import warnings  
warnings.filterwarnings("ignore")  
import math

**Normal Distribution Confidence Interval**

def norm\_ci(confidence\_interval: float = 0.90,  
 population\_std\_dev: float = 1,  
 number\_of\_samples:int = 1,  
 mean: float = 0,  
 two\_sided: bool = True  
 ):  
 '''  
   
  
 Parameters  
 ----------  
 confidence\_interval : float, optional  
 DESCRIPTION. The default is 0.90.  
 population\_std\_dev : float, optional  
 DESCRIPTION. The default is 1.  
 number\_of\_samples : int, optional  
 DESCRIPTION. The default is 100.  
 mean : float, optional  
 DESCRIPTION. The default is 0.  
 two\_sided : bool, optional  
 DESCRIPTION. The default is True.  
  
 Returns  
 -------  
 TYPE  
 DESCRIPTION.  
  
 '''  
 # declaring problem constants  
 x\_bar = mean  
 n = number\_of\_samples # number of samples taken  
 sigma = population\_std\_dev # population standard deviation  
 sigma\_x\_bar = sigma/n\*\*0.5 # std dev of sample means  
   
 # generate x within 3.5 standard deviations and y axes  
 x\_axis\_lower\_bound = x\_bar-3.5\*sigma\_x\_bar  
 x\_axis\_upper\_bound = x\_bar+3.5\*sigma\_x\_bar  
 steps = (x\_axis\_upper\_bound-x\_axis\_lower\_bound)/1000  
 x = np.arange(x\_axis\_lower\_bound,x\_axis\_upper\_bound,steps)  
 y = norm.pdf(x,loc=x\_bar,scale=sigma\_x\_bar)  
   
 # get the probabilities of the tail areas  
 if two\_sided:  
 alpha\_high = (1+confidence\_interval)/2  
 alpha\_low = (1-confidence\_interval)/2  
 else:  
 alpha\_high = confidence\_interval  
 alpha\_low = 0  
 # compute the value of x\_lower and x\_higher  
 x\_lower = norm.ppf(alpha\_low,loc=x\_bar,scale=sigma\_x\_bar)  
 x\_higher = norm.ppf(alpha\_high,loc=x\_bar,scale=sigma\_x\_bar)  
   
 # plot and shade the graphs  
 plt.plot(x,y,'-')  
 plt.fill\_between(x[x>=x\_lower],y[x>=x\_lower],color='red')  
 plt.fill\_between(x[x>=x\_higher],y[x>=x\_higher],color='white')  
 plt.draw()  
 return round(x\_lower,3), round(x\_higher,3)

**t- distribution Confidence Interval**

def t\_ci(confidence\_interval: float = 0.90,  
 population\_std\_dev: float = 1,  
 number\_of\_samples:int = 1,  
 mean: float = 0,  
 two\_sided: bool = True   
  
 ):  
   
 #declaring problem constants  
 x\_bar = mean  
 n = number\_of\_samples # number of samples taken  
 sigma = sample\_std\_dev # population standard deviation  
 sigma\_x\_bar = sigma/n\*0.5 # std dev of sample means  
 df = n-1 # degrees of freedom  
   
   
 # get the probabilities of the tail areas  
 if two\_sided:  
 alpha\_high = (1+confidence\_interval)/2  
 alpha\_low = (1-confidence\_interval)/2  
 else:  
 alpha\_high = confidence\_interval  
 alpha\_low = 0  
 # compute the value of x\_lower and x\_higher  
 x\_lower = t.ppf(alpha\_low,df,loc=x\_bar,scale=sigma\_x\_bar)  
 x\_higher = t.ppf(alpha\_high,df,loc=x\_bar,scale=sigma\_x\_bar)  
   
 # plot and shade the graphs  
 # plt.plot(x,y,'-')  
 # plt.fill\_between(x[x>=x\_lower],y[x>=x\_lower],color='green')  
 # plt.fill\_between(x[x>=x\_higher],y[x>=x\_higher],color='white')  
 # plt.draw()  
 return round(x\_lower,3), round(x\_higher,3)

**Reading the Dataset**

df = pd.read\_csv(r'SA1\_Group\_17.csv', index\_col='Index')

**Question:1**

confidence\_interval = 0.95  
sample\_std\_dev = df.GOP\_Year3.std()  
number\_of\_samples = len(df.GOP\_Year3)  
sample\_mean = df.GOP\_Year3.mean()  
lower, higher = t\_ci(confidence\_interval,sample\_std\_dev,number\_of\_samples,sample\_mean)  
print('Mean of Gross output – Year 3 of population is expected to lie between Rs. {} and Rs. {}'.format(lower,higher))

🡺Mean of Gross output – Year 3 of population is expected to lie between Rs. 91700301.465 and Rs. 93480369.722

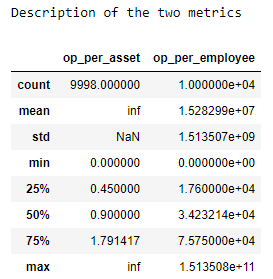
**Question 2:**

Defining metrics for performance of the units We define the performance of the units as follows:

op\_per\_asset = GOP\_Year3/MKT\_VAL\_FA. This metric is useful in determining how the units are performing on the basis of utilization of the fixed assets. As a basic understanding more the MKT\_VAL\_FA more should be GOP\_Year3. If the ratio is low for any unit it means there might be a problem of under utilization of resources happening in that given unit. Also if the ratio is too high denotes the units are working with highly deprecated assets which can be a great risk sooner or later.

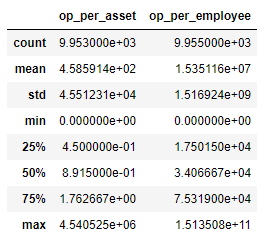
op\_per\_employee: GOP\_Year3/EMP\_TOTAL In these world of automation initiatives to increase productivitiy of business this metric is very useful. If the ratio is too low it means those units might potentially show redundancies in job roles. Employees of those units might be available to take up newer challenging roles which in turn will be increasing the business. Units showing too high value might be facing employee shortage problems.

df['op\_per\_asset'] = df['GOP\_Year3']/df['MKT\_VAL\_FA']  
df['op\_per\_employee'] = df['GOP\_Year3']/df['EMP\_TOTAL']  
print('Description of the two metrics')  
display(df[['op\_per\_asset','op\_per\_employee']].describe())



🡺Inference: It seems there are some rows in the data where the MKT\_VAL\_FA = 0 due to which we are getting inf values. This seems to be some sort of a data collection issue. To avoid this we filter the data to remove the inf values. Also there seems to be some missing values too.

import math  
filtered\_df = df[~df.op\_per\_asset.isna()]  
filtered\_df = df[df.op\_per\_asset != math.inf]  
filtered\_df[['op\_per\_asset','op\_per\_employee']].describe()



🡺Inference: This resulted in losing out around 40 data points from our sample. There can be a separate analysis how those 40 data points had MKT\_VAL\_FA = 0. But this is out of scope for this exercise.

**Question3:**

🡺99% confidence interval for op\_per\_asset: Since we have described that both low op\_per\_asset and high op\_per\_asset is a problem here, we will define two sided confidence interval for the given metric

confidence\_interval = 0.99  
sample\_std\_dev = filtered\_df.op\_per\_asset.std()  
number\_of\_samples = len(filtered\_df.op\_per\_asset)  
sample\_mean = filtered\_df.op\_per\_asset.mean()  
two\_sided = True  
lower, higher = t\_ci(confidence\_interval,  
 sample\_std\_dev,  
 number\_of\_samples,  
 sample\_mean,  
 two\_sided  
 )  
print('Mean of Output Per Asset as of Year 3 of population is expected to lie between {} and {}'.format(lower,higher))

Mean of Output Per Asset as of Year 3 of population is expected to lie between 452.702 and 464.481

🡺99% confidence interval for op\_per\_employee: Since we have described that both low op\_per\_employee and high op\_per\_asset is a problem here, we will define two sided confidence interval for the given metric

confidence\_interval = 0.99  
sample\_std\_dev = filtered\_df.op\_per\_employee.std()  
number\_of\_samples = len(filtered\_df.op\_per\_employee)  
sample\_mean = filtered\_df.op\_per\_employee.mean()  
two\_sided = True  
lower, higher = t\_ci(confidence\_interval,  
 sample\_std\_dev,  
 number\_of\_samples,  
 sample\_mean,  
 two\_sided  
 )  
print('Mean of Gross Output Per Employee as of Year 3 of population is expected to lie between Rs.{} and Rs.{}'.format(lower,higher))

🡺Mean of Gross Output Per Employee as of Year 3 of population is expected to lie between Rs.15154870.774 and Rs.15547446.054

**Question4:**

**a)**Probability that a firm selected at random is a SSSBE unit filter only SSSBE units

p = len(df[df.UNIT\_TYPE==2])/len(df)  
print(f'Probability = {round(p,3)}')

Probability = 0.216

**b)**Probability that a firm selected at random is GOOD in performance We calculate this by checking if the values of the column op\_per\_asset > mean of the column op\_per\_asset

mean\_op\_per\_asset = filtered\_df.op\_per\_asset.mean()  
filtered\_df['good\_in\_performance'] = filtered\_df.op\_per\_asset.apply(lambda row: row > mean\_op\_per\_asset)  
p = len(filtered\_df[filtered\_df.good\_in\_performance==True])/len(filtered\_df)  
print(f'Probability = {round(p,10)}')  
print('Number of units performing good = {}'.format(len(filtered\_df[filtered\_df.good\_in\_performance==True])))

Probability = 0.0003013561  
Number of units performing good = 3

**c)**Probability that a firm selected is a SSSBE Unit and ALSO GOOD in performance

n\_sssbe\_good\_performance = len(filtered\_df[(filtered\_df.UNIT\_TYPE==2) \  
 &(filtered\_df.good\_in\_performance==True) \  
 ])  
print('Probability that firm is SSSBE Unit and also a good performer = {0:.9f}'.format(n\_sssbe\_good\_performance/len(filtered\_df)))  
p\_good\_given\_sssbe = n\_sssbe\_good\_performance/len(filtered\_df[(filtered\_df.UNIT\_TYPE==2)])  
print('Conditional probability that a firm is Good given that its SSSBE:{}'.format(p\_good\_given\_sssbe))

Probability that firm is SSSBE Unit and also a good performer = 0.000200904  
Conditional probability that a firm is Good given that its SSSBE:0.0009280742459396752

**d)**Conclusion about performance of SSSBE units From calculation in 4c. we can see that only a mere 0.02% of our sample data comprise of performances from SSSBE units which are performing good. But to conclude whether SSSBE units performance are good or bad in compared to SSI we have to do a comparative study.

n\_ssi\_good\_performance = len(filtered\_df[(filtered\_df.UNIT\_TYPE==1) \  
 &(filtered\_df.good\_in\_performance==True) \  
 ])  
print('Probability that firm is SSI Unit and also a good performer = {0:.9f}'.format(n\_ssi\_good\_performance/len(filtered\_df)))  
p\_good\_given\_ssi = n\_ssi\_good\_performance/len(filtered\_df[(filtered\_df.UNIT\_TYPE==1)])  
print('Conditional probability that a firm is Good given that its SSI:{}'.format(p\_good\_given\_ssi))

Probability that firm is SSI Unit and also a good performer = 0.000100452  
Conditional probability that a firm is Good given that its SSI:0.0001282051282051282

🡺Inference: From the above calculations it is clear that:

A majority of good performer is SSBE units and not SSI units in our sample. If we see the conditional probability to understand if given that a firm is an SSSBE unit what is the probability that it will perform good < if given that a firm is SSI Unit what is the probability of being good performer. Based on these above caclculations it is evident that performance of SSI unit is not good as compared to SSSBE Units.

**Question:5**

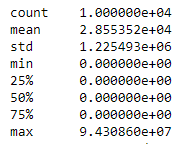
Null Hypothesis test

Null hypothesis H0: Population mean of VOE\_Year3 = 87,300.

Alternate Hypothesis H1: Population mean of VOE\_Year3 ≠ 87,300

We will setup a one sided confidence interval of 0.95 Since population standard deviation is not given to us we will use sample standard deviation and use t test

df.VOE\_Year3.describe()



confidence\_interval = 0.95  
sample\_std\_dev = df.VOE\_Year3.std()  
number\_of\_samples = len(df.VOE\_Year3)  
mean = 87300  
two\_sided = False  
lower, higher = t\_ci(confidence\_interval,sample\_std\_dev,number\_of\_samples,mean,two\_sided)  
sample\_mean = df.VOE\_Year3.mean()  
print('Sample Mean for Value of Exports for Year 3 is expected to lie between Rs. {} and Rs. {}'.format(lower,higher))  
print(f'Does sample mean falls within above range? Ans: {lower<=sample\_mean<=higher} and the value {sample\_mean}')  
print(f'The t value for sample mean:{t.cdf(sample\_mean,df=number\_of\_samples-1,loc=mean,scale=sample\_std\_dev/number\_of\_samples\*\*0.5)}')

Sample Mean for Value of Exports for Year 3 is expected to lie between Rs. -inf and Rs. 87400.797  
Does sample mean falls within above range? Ans: True and the value 28553.5212  
The t value for sample mean:8.304663866157408e-07

🡺Inference: It is evident that from the one sided t-test though the sample mean lies between the given ranges the P value is << 0.05. Hence we can surely reject the Null hypothesis that the population mean of VOE\_Year3 is 87300.

**Question:6**

Special incentives for SSSBE or SSI or both Explanation: Below we will define the success criteria as follows: If unit is an SSSBE Unit its a success. We calculate the population proportion of its success rate. If unit is an SSI Unit its a success. We calculate the population proportion of its success rate. For the unit to get incentives the population proportion of it should be < 0.25

import statsmodels.api as sm  
from statsmodels.stats.proportion import proportion\_confint

confidence\_interval = 0.99  
sssbe\_count = len(df[df.UNIT\_TYPE==2])  
total\_count = len(df)  
  
  
sssbe\_pop\_prop = proportion\_confint(count=sssbe\_count, # Number of "successes"  
 nobs=total\_count, # Number of trials  
 alpha=(1 - confidence\_interval))  
print('Population proportion of SSSBE units is expected to lie within {} by confidence interval of {}'.format(sssbe\_pop\_prop, confidence\_interval))

Population proportion of SSSBE units is expected to lie within (0.2051054318328597, 0.2262945681671403) by confidence interval of 0.99

confidence\_interval = 0.99  
sssbe\_count = len(df[df.UNIT\_TYPE==1])  
total\_count = len(df)  
  
sssbe\_pop\_prop = proportion\_confint(count=sssbe\_count, # Number of "successes"  
 nobs=total\_count, # Number of trials  
 alpha=(1 - confidence\_interval))  
print('Population proportion of SSI units is expected to lie within {} by confidence interval of {}'.format(sssbe\_pop\_prop, confidence\_interval))

Population proportion of SSI units is expected to lie within (0.7737054318328597, 0.7948945681671403) by confidence interval of 0.99

Inference: Since SSSBE Unit's population proportion is expected to be lying below 25% we would recommend these special incentives for SSSBE.

**Question:7**

Contention that a larger proportion of SSSBEs are managed by men as compared to women Explanation: For this we will estimate population proportion of SSSBE Units managed by Male. The column MAN\_BY will be beneficial for this case.

**1)**We define success if a unit is managed by man. 2.We estimate the population proportion of SSSBE units to be managed by men from our sample by a set confidence interval. 3.If the estimated population proportion > 0.5 this contention will hold true. filter out only SSSBE Units

sssbe\_df = df[df.UNIT\_TYPE == 2]  
sssbe\_df.MAN\_BY.value\_counts()

1 2102  
2 55  
Name: MAN\_BY, dtype: int64

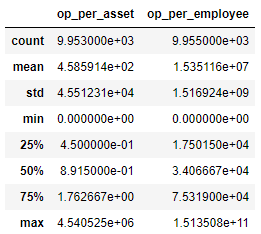
confidence\_interval = 0.99  
no\_of\_sssbe\_units\_managed\_by\_men = len(sssbe\_df[sssbe\_df.MAN\_BY == 1])  
number\_of\_sssbe\_units = len(sssbe\_df)  
male\_employee\_pop\_prop = proportion\_confint(count=no\_of\_sssbe\_units\_managed\_by\_men, # Number of "successes"  
 nobs=number\_of\_sssbe\_units, # Number of trials  
 alpha=(1 - confidence\_interval))  
print('Population proportion of SSSBE units being managed by men is expected to lie within {} by confidence interval of {}'.format(sssbe\_pop\_prop, confidence\_interval))

Population proportion of SSSBE units being managed by men is expected to lie within (0.7737054318328597, 0.7948945681671403) by confidence interval of 0.99

**Question:8**

Distribution of defined metrics

filtered\_df[['op\_per\_asset','op\_per\_employee']].describe()



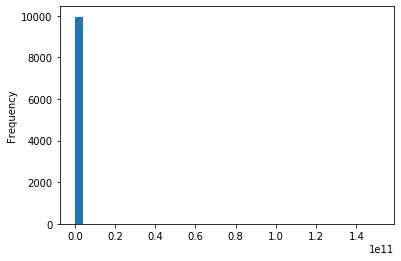
filtered\_df['op\_per\_asset'].plot.hist(bins=35)

<matplotlib.axes.\_subplots.AxesSubplot at 0x2b381316e48>



filtered\_df['op\_per\_employee'].plot.hist(bins=35)

<matplotlib.axes.\_subplots.AxesSubplot at 0x2b381fed3c8>

Inference: The distributions of the metrics 'op\_per\_asset' and 'op\_per\_employee' are right skewed in nature. We can find the evidence from the above histograms and also the .describe() method here where it is seen that median << mean