#Name : B.Harini Shree

#Dept : CSE

#Roll No. : 37

#ELECTRICITY DATA SET ANALYSIS

#Description:

#Energy consumption is proportional to the rise in population. A dataset from https://data.gov.in/ is taken. The dataset is classified into 5 sectors namely (domestic, commercial, industrial, public usage and others). An analysis is made to check the highest electricity consuming sector of the economy.

#Statistical information on quantitative variables

In [1]:

```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy.stats import t
import random
from collections import Counter
```

In [13]:

```
data = pd.read_csv("C:/Users/B.MALATHI/Desktop/Ind.csv")
data.head(5)
```

Out[13]:

	Dates	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP	J&K	Chandigarh	...	Odisha	West Bengal	Sikkim	Arunac Prade
0	2019-02-01	119.9	130.3	234.1	85.8	313.9	40.7	30.0	52.5	5.0	...	70.2	108.2	2.0	
1	2019-03-01	121.9	133.5	240.2	85.5	311.8	39.3	30.1	54.1	4.9	...	67.9	110.2	1.9	
2	2019-04-01	118.8	128.2	239.8	83.5	320.7	38.1	30.1	53.2	4.8	...	66.3	106.8	1.7	
3	2019-05-01	121.0	127.5	239.1	79.2	299.0	39.2	30.2	51.5	4.3	...	65.8	107.0	2.0	
4	2019-06-01	121.4	132.6	240.4	76.6	286.8	39.2	31.0	53.2	4.3	...	62.9	106.4	2.0	

5 rows × 34 columns

In [14]:

```
data.isnull().sum()
```

Out[14]:

```
Dates                  0
Punjab                 0
Haryana                0
Rajasthan              0
Delhi                  0
UP                     0
Uttarakhand            0
HP                     0
J&K                    0
Chandigarh             0
Chhattisgarh           0
Gujarat                0
MP                     0
Maharashtra            0
Goa                    0
DNH                    0
Andhra Pradesh         0
Telangana              0
Karnataka              0
Kerala                 0
Tamil Nadu             0
Pondy                  0
Bihar                  0
Jharkhand              0
Odisha                 0
West Bengal            0
Sikkim                 0
Arunachal Pradesh      0
Assam                  0
Manipur                0
Meghalaya              0
Mizoram                0
Nagaland               0
Tripura                0
dtype: int64
```

In [15]:

```
data.shape
```

Out[15]:

```
(503, 34)
```

In [16]:

```
data.describe()
```

Out[16]:

	Punjab	Haryana	Rajasthan	Delhi	UP	Uttarakhand	HP	J&K	Chandigarh	Chhatti
count	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000	503.0
mean	141.145527	138.333598	218.443340	83.380716	314.036382	36.157058	26.568191	44.264016	4.141551	83.8
std	56.977361	38.106593	27.421615	25.915357	66.516960	6.705108	4.807040	4.769391	1.143422	10.1
min	56.100000	64.800000	105.800000	41.800000	186.800000	16.800000	11.800000	17.800000	2.200000	37.2
25%	104.000000	114.800000	205.800000	63.500000	263.650000	33.800000	25.600000	41.550000	3.300000	75.7
50%	118.300000	126.800000	222.900000	72.700000	290.000000	37.000000	28.000000	44.100000	3.800000	82.6
75%	162.500000	158.100000	237.600000	105.800000	370.550000	40.350000	29.700000	47.350000	4.900000	91.6
max	300.000000	237.200000	278.000000	147.100000	471.800000	53.200000	34.000000	54.200000	7.400000	111.6

8 rows × 33 columns

#Sample

In [31]:

```
y = data['Punjab']

y.describe()
```

Out[31]:

```
count    503.000000
mean     141.145527
std       56.977361
min       56.100000
25%      104.000000
50%      118.300000
75%      162.500000
max      300.000000
Name: Punjab, dtype: float64
```

In [32]:

```
y.mean()
```

Out[32]:

```
141.1455268389662
```

In [33]:

```
y.std()
```

Out[33]:

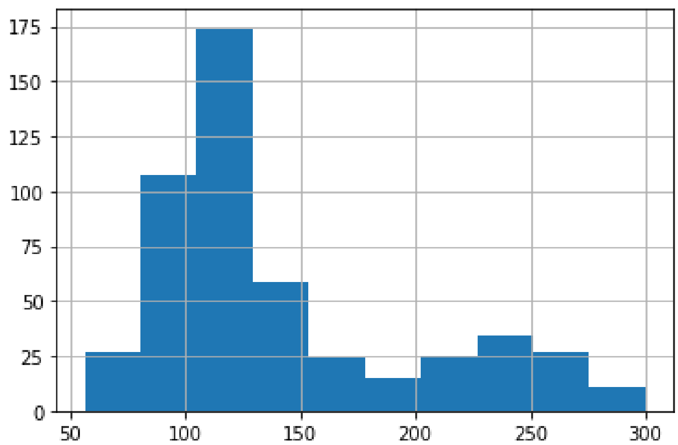
```
56.977360560183484
```

In [34]:

```
y.hist()
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x224595386a0>
```



In [35]:

```
a = y.sample(n=50)
```

In [36]:

```
a.describe()
```

Out[36]:

Out[36]:

```
count     50.000000
mean     136.130000
std       51.901513
min       68.500000
25%      103.500000
50%      116.650000
75%      162.300000
max      291.700000
Name: Punjab, dtype: float64
```

In [37]:

```
a.mean()
```

Out[37]:

```
136.13
```

In [38]:

```
a.std()
```

Out[38]:

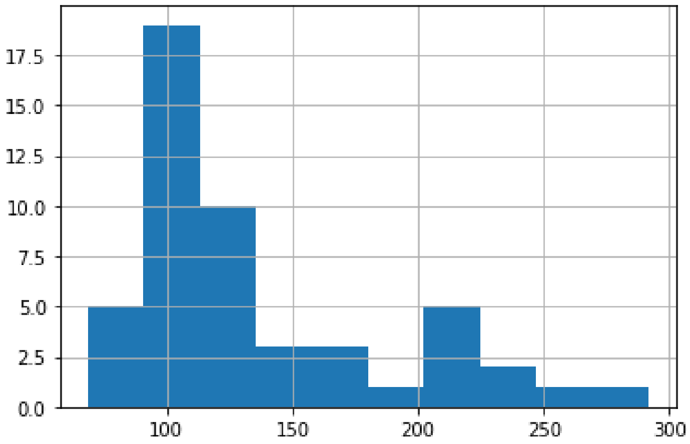
```
51.90151289525506
```

In [39]:

```
a.hist()
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22459508cc0>
```



In [40]:

```
b = y.sample(n=100)
```

In [41]:

```
b.describe()
```

Out[41]:

```
count    100.000000
mean     142.881000
std       58.019227
min       69.500000
25%      105.625000
50%      116.550000
75%      158.425000
max      291.700000
Name: Punjab, dtype: float64
```

In [42]:

```
b.mean()
```

Out[42]:

```
142.88100000000003
```

In [43]:

```
b.std()
```

Out[43]:

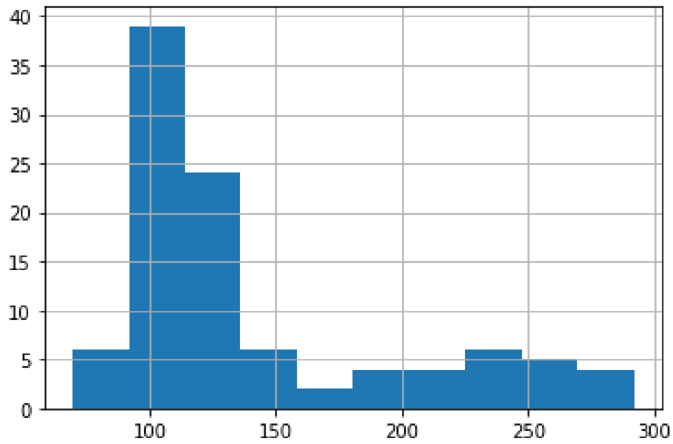
```
58.019226515479716
```

In [44]:

```
b.hist()
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x224594d19e8>
```



#SAMPLE DISTRIBUTION ON QUANTITAIVE VARIABLE

In [46]:

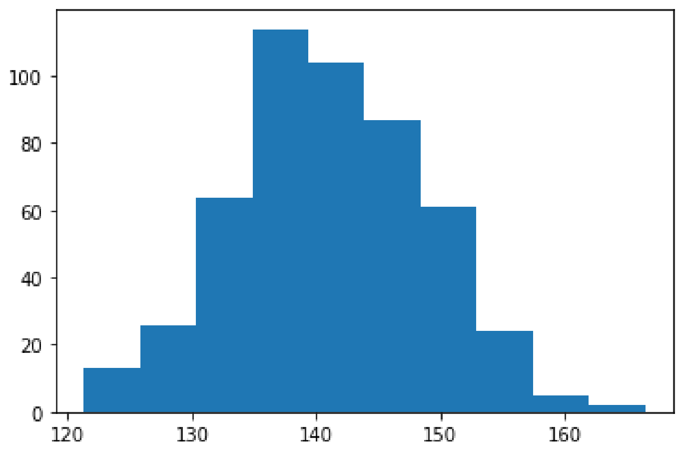
```
sample_means = np.repeat(np.nan,500)
for i in range(500):
    sample = y.sample(n = 50)
    sample_means[i] = sample.mean()
```

In [47]:

```
plt.hist(sample_means)
```

Out[47]:

```
(array([ 13.,  26.,  64., 114., 104.,  87.,  61.,  24.,   5.,   2.]),
 array([121.31   , 125.8224, 130.3348, 134.8472, 139.3596, 143.872 ,
        148.3844, 152.8968, 157.4092, 161.9216, 166.434 ]),
 <a list of 10 Patch objects>)
```



In [48]:

```
sample_means.mean()
```

Out[48]:

```
141.08678
```

In [49]:

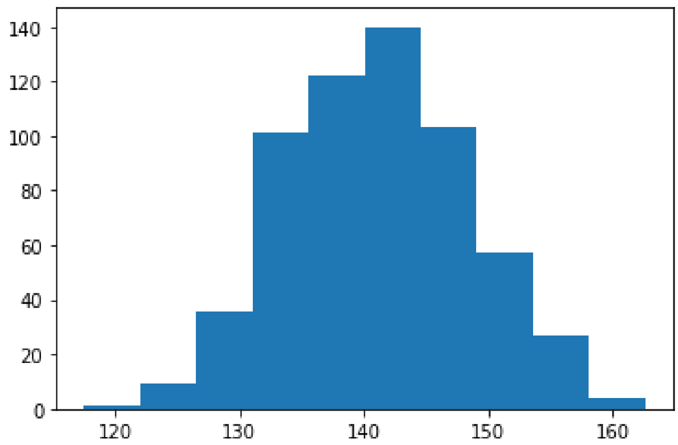
```
sample_means = np.repeat(np.nan,600)
for i in range(600):
    sample = y.sample(n = 50)
    sample_means[i] = sample.mean()
```

In [50]:

```
plt.hist(sample_means)
```

Out[50]:

```
(array([  1.,   9.,  36., 101., 122., 140., 103.,  57.,  27.,   4.]),
 array([117.516 , 122.0258, 126.5356, 131.0454, 135.5552, 140.065 ,
        144.5748, 149.0846, 153.5944, 158.1042, 162.614 ]),
 <a list of 10 Patch objects>)
```



In [51]:

```
sample_means.mean()
```

Out[51]:

```
141.19878666666665
```

#Population mean and population standard deviation

In [52]:

```
y = data['Punjab']
```

In [53]:

```
pop_mean = y.mean()
pop_std = y.std()
```

In [54]:

```
print("Population_mean", pop_mean)
print("Population_std", pop_std)
```

```
Population_mean 141.1455268389662
Population_std 56.977360560183484
```

In [55]:

```
c = y.sample(n=50)
```

In [56]:

```
c.describe()
```

Out[56]:

```
count     50.000000
mean     146.050000
std       61.208284
min       63.100000
25%      104.775000
50%      121.100000
75%      159.375000
max      291.700000
Name: Punjab, dtype: float64
```

In [57]:

```
sample_mean = c.mean()
sample_std = c.std()
```

In [58]:

```
print("Sample_mean", sample_mean)
print("Sample_std", sample_std)
```

```
Sample_mean 146.05
Sample_std 61.20828358638915
```

#Interval estimate for known Population Standard deviation

In [59]:

```
n = 100
```

In [61]:

```
z_score = norm.ppf(0.924)
z_score
```

Out[61]:

```
1.432502720825812
```

In [62]:

```
std_error = pop_std/math.sqrt(n)
mar_error = z_score * std_error
```

In [63]:

```
print("Std Error", std_error)
print("Mar Error", mar_error)
```

```
Std Error 5.697736056018348
Mar Error 8.162022402793616
```

In [64]:

```
ie1 = sample_mean + mar_error
ie2 = sample_mean - mar_error
```

In [65]:

```
(ie1, ie2)
```

Out[65]:

```
(154.21202240279362, 137.8879775972064)
```

In [67]:

```
pop_mean
```

Out[67]:

```
141.1455268389662
```

#Interval estimate for unknown Population Standard deviation

In [68]:

```
n = 100
```

In [69]:

```
t_score = t.ppf(q=0.924, df=n-1)
t_score
```

Out[69]:

```
1.443630448591297
```

In [70]:

```
std_error = pop_std/math.sqrt(n)
mar_error = t_score * std_error
```

In [71]:

```
print("Std Error", std_error)
print("Mar Error", mar_error)
```

```
Std Error 5.697736056018348
Mar Error 8.225425258504576
```

In [72]:

```
ie1 = sample_mean + mar_error
ie2 = sample_mean - mar_error
```

In [73]:

```
(ie1, ie2)
```

Out[73]:

```
(154.27542525850458, 137.82457474149544)
```

#Two tailed Hypothesis testing

In [74]:

```
total = sum(data['Punjab'])
length = len(data['Punjab'])

avg = total / length
avg
```

Out[74]:

```
141.1455268389662
```

#Hypothesis

In [78]:

```
from scipy.stats import ttest_1samp
a = data['Punjab']
tset, pval = ttest_1samp(a, 141)
```

In [79]:

```
print("P-Values:", pval)
```

```
P-Values: 0.9543426595426494
```

In [80]:

```
if pval < 0.05:
    print("We are rejecting null hypothesis")
else:
    print("We are accepting null hypothesis")
```

```
We are accepting null hypothesis
```

In [81]:

```
a.mean()
```

Out[81]:

```
141.1455268389662
```

#Bootstrap resampling

In [82]:

```
y = data['Punjab']
sample_draw = y.sample(n=50)
```

In [83]:

```
n = len(sample_draw)
b = 10000
```

In [84]:

```
sample_means = np.repeat(np.nan,10000)
for i in range(10000):
    sample = y.sample(n = 40)
    sample_means[i] = sample.mean()
```

In [85]:

```
boot_mean = sample_means.mean()
boot_mean
```

Out[85]:

```
140.97684375
```

#Qualitative variable

In [86]:

```
data = pd.read_csv("C:/Users/B.MALATHI/Desktop/Iris.csv")
data.head(5)
```

Out[86]:

Out[86]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [87]:

```
data.describe()
```

Out[87]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [88]:

```
data.isnull().sum()
```

Out[88]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [89]:

```
data.shape
```

Out[89]:

```
(150, 6)
```

#Sample

In [92]:

```
y = Counter(data['Species'])
y
```

Out[92]:

```
Counter({'Iris-setosa': 50, 'Iris-versicolor': 50, 'Iris-virginica': 50})
```

In [93]:

```
pop_prop = y['Iris-setosa'] / 150
```

```
pop_prop
```

Out[93]:

```
0.3333333333333333
```

In [102]:

```
samp_prop = y['Iris-setosa'] / 50
samp_prop
```

Out[102]:

```
1.0
```

In [103]:

```
std_error = math.sqrt(pop_prop * (1-pop_prop)/50)
mar_error = norm.ppf(0.924) * std_error
```

In [104]:

```
print("Std Error", std_error)
print("Mar Error", mar_error)
```

```
Std Error 0.06666666666666667
Mar Error 0.09550018138838748
```

In [109]:

```
ie1 = samp_prop + mar_error
ie2 = samp_prop - mar_error
```

In [110]:

```
(ie1, ie2)
```

Out[110]:

```
(1.0955001813883876, 0.9044998186116125)
```

In [ ]:

```

```