In [38]:
```
from  scipy.stats import binom,poisson,expon,mode
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
import math
warnings.filterwarnings("ignore")

### source: tinyurl.com/4zs92xtk
```

Assuming that X represents the data (population), if X has a distribution with average μ and standard deviation σ, and if X is approximately normally distributed or if the sample size n is large.

Then the sample mean $\overline{X}$ is normally distributed with the avarage μ and standard error σ/sqrt(n)

The above Distribution is only valid if,

X is approximately normal or sample size n is large, and,

the data (population) standard deviation σ is known.

If X is normal, then $\overline{X}$ is also normally distributed regardless of the sample size n. Central Limit Theorem tells us that even if X is not normal, if the sample size is large enough (usually greater than 30), then $\overline{X}$'s distribution is approximately normal (Sharpe, De Veaux, Velleman and Wright, 2020, pp. 318–320). If $\overline{X}$ is normal, we can easily standardize and convert it to the standard normal distribution Z.

If the population standard deviation σ is not known, we cannot assume that the sample mean $\overline{X}$ is normally distributed. If certain conditions are satisfied (explained below), then we can transform $\overline{X}$ to another random variable t such that,

$$t = \frac{\overline{X} - \mu}{s/\sqrt{n}}$$

The random variable t is said to follow the t-distribution with n-1 degrees of freedom, where n is the sample size. The t-distribution is bell-shaped and symmetric (just like the normal distribution) but has fatter tails compared to the normal distribution. This means values further away from the mean have a higher likelihood of occurring compared to that in the normal distribution.

The conditions to use the t-distribution for the random variable t are as follows (Sharpe et al., 2020, pp. 415–420):

If X is normally distributed, even for small sample sizes (n<15), the t-distribution can be used.

If the sample size is between 15 and 40, the t-distribution can be used as long as X is unimodal and reasonably symmetric. For sample sizes greater than 40, the t-distribution can be used unless X's distribution is heavily skewed.

Let's draw a sample of size n=250 from the normal distribution. Here we are assuming that our data is normally distributed and has parameters μ = 20 and σ = 3. Collecting one sample from this population

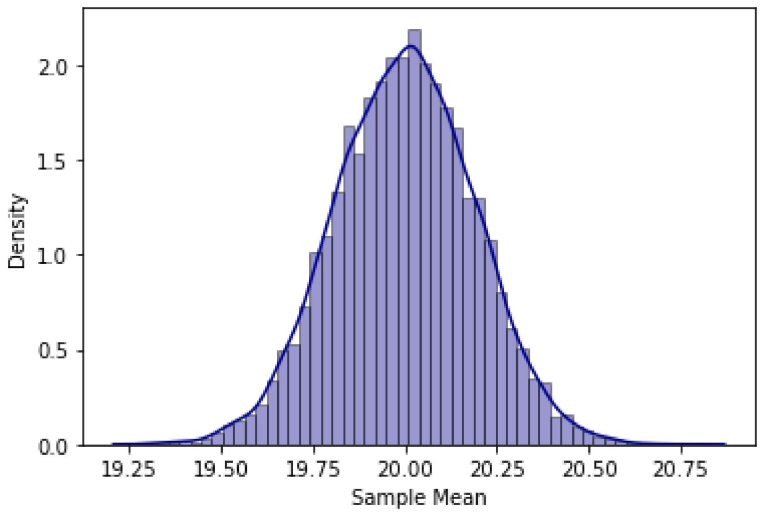
In [6]:
```
mu, sigma = 20, 3
sample = np.random.normal(mu, sigma, 250)
x_bar = sample.mean()
x_bar
```

Out[6]:
```
20.042507169222105
```

In [7]:
```
## But if I ran this code 10,000 times and recorded the values of x and plot

import seaborn as sns
samples = np.zeros(10000)
mu, sigma = 20, 3
for s in range(10000):
    sample = np.random.normal(mu, sigma, 250)  #sample of size 250
    x_bar = sample.mean()                      #calculating sample mean
    samples[s] = x_bar
sns.distplot(samples, hist = True, color = 'darkblue',
             hist_kws={'edgecolor':'black'}).set(xlabel = 'Sample Mean',
                                                 ylabel = 'Density')
```

Out[7]:
```
[Text(0.5, 0, 'Sample Mean'), Text(0, 0.5, 'Density')]
```



As you can see, the distribution is approximately symmetric and bell-shaped (just like the normal distribution) with an average of approximately 20 and a standard error that is approximately equal to 3/sqrt(250) = 0.19.

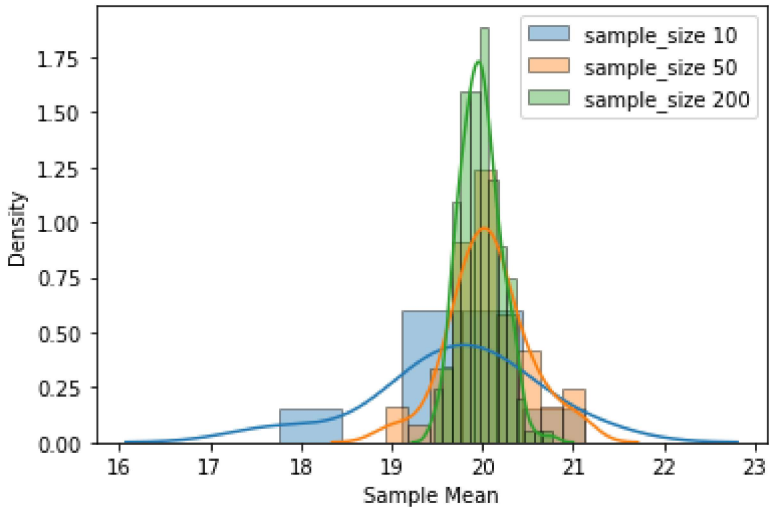
Sampling from the same population with different sample sizes will result in different measures of spread in the outcome distribution. As we expect, increasing the sample size will reduce the standard error and therefore, the distribution will be narrower around its average. Note that the

distribution of $\bar{X}$ is normal even for extremely small sample sizes. This is because X is normally distributed.

In [17]:
```
mu, sigma = 20, 3

for sample_size in [10,50,200]:
    samples = np.zeros(sample_size)
    for s in range(sample_size):
        sample = np.random.normal(mu, sigma, sample_size)  #sample of size 2
        x_bar = sample.mean()                              #calculating sample mean
        samples[s] = x_bar
    sns.distplot(samples, hist = True,
                 hist_kws={'edgecolor':'black'},label= 'sample_size '+ str(s
                                                ylabel = 'Density')
plt.legend()
```

Out[17]:
```
<matplotlib.legend.Legend at 0x21d2287ca00>
```

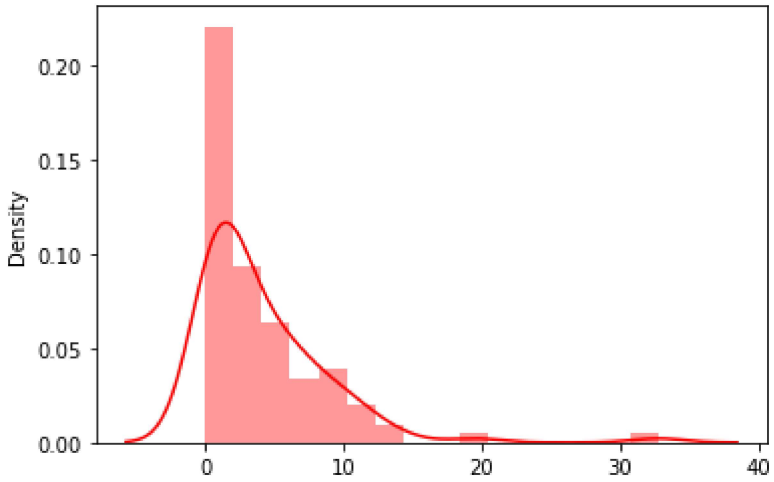


What if the population (data) is not normal?

No worries! Even if your data is not normally distributed, if the sample size is large enough, the distribution of $\bar{X}$ can still be approximated using the normal distribution (according to Central Limit Theorem). The following figure shows the distribution of $\bar{X}$ when X is heavily skewed to the left. As you can see, $\bar{X}$'s distribution tends to mimic the distribution of X for small sample sizes. However, as sample size grows the distribution of $\bar{X}$ becomes more symmetric and bell-shaped. As mentioned above, if sample size is large (usually larger than 30), $\bar{X}$'s distribution is approximately normal regardless of what the distribution of X is.

In [18]:
```
### Lets sample from exponential distribution and understand how the mean of
expon_data = expon.rvs(scale=4,size= 100)
sns.distplot(expon_data ,hist=True,kde=True,color="red")
```

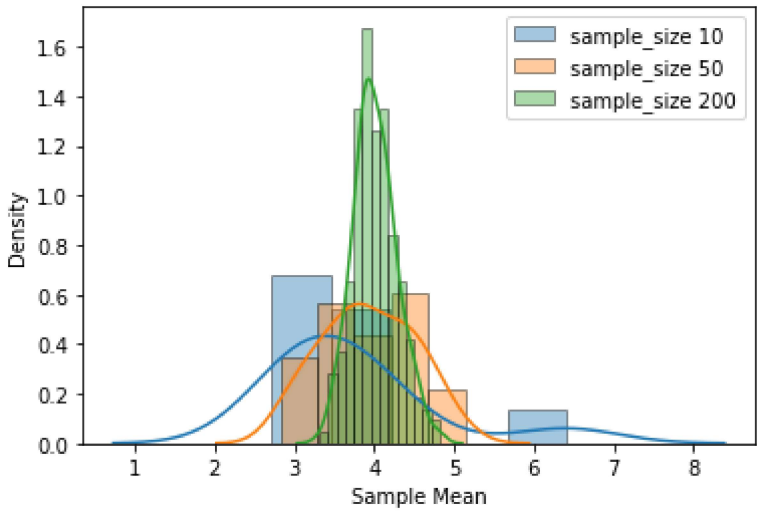
Out[18]: <AxesSubplot:ylabel='Density'>



In [19]:
```
mu, sigma = 20, 3

for sample_size in [10,50,200]:
    samples = np.zeros(sample_size)
    for s in range(sample_size):
        sample = expon.rvs(scale=4,size= sample_size) #sample of size 250
        x_bar = sample.mean()                         #calculating sample mean
        samples[s] = x_bar
    sns.distplot(samples, hist = True,
                 hist_kws={'edgecolor':'black'},label= 'sample_size '+ str(s
                                                ylabel = 'Density')
plt.legend()
```

Out[19]: <matplotlib.legend.Legend at 0x21d2192e280>



As we can see if we increase the sample size the distribution come's down to a normal distribution with mean 4

What happens when we compare the median sample distribution with mean sample distribution

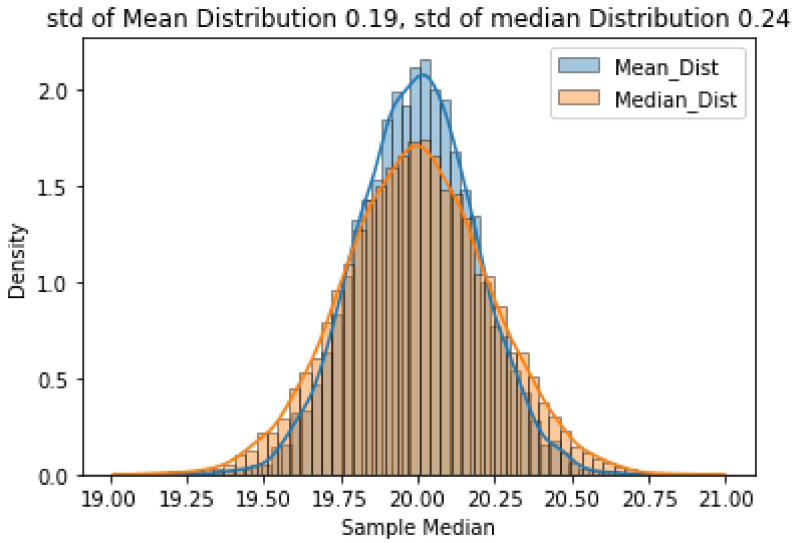
In [33]:

```
## But if I ran this code 10,000 times and recorded the values of x and plot

import seaborn as sns
samples_mean = np.zeros(10000)
samples_median = np.zeros(10000)
mu, sigma = 20, 3
for s in range(10000):
    sample = np.random.normal(mu, sigma, 250)  #sample of size 250
    x_bar = sample.mean()                      #calculating sample mean
    samples_mean[s] = x_bar
    samples_median[s] = np.median(sample)
sns.distplot(samples_mean, hist = True, label = 'Mean_Dist',
             hist_kws={'edgecolor':'black'}).set(xlabel = 'Sample Mean',
                                                 ylabel = 'Density')
sns.distplot(samples_median, hist = True, label = 'Median_Dist',
             hist_kws={'edgecolor':'black'}).set(xlabel = 'Sample Median',
                                                 ylabel = 'Density')
plt.title("std of Mean Distribution {}, std of median Distribution {}".forma

plt.legend()
```

Out[33]: <matplotlib.legend.Legend at 0x21d2874fe20>



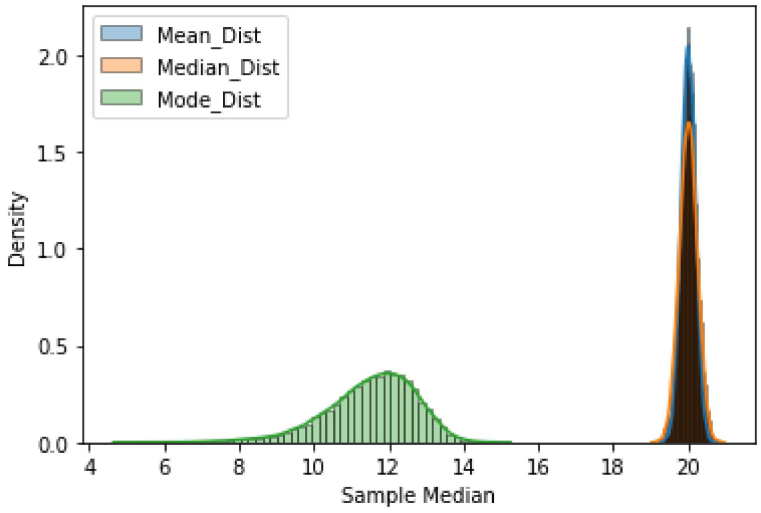
As we can see mean is same for Distribution of sample Mean and Median, but standard deviation is more for Median by .24/.19 around 1.26

Lets check what happens with the mode

In [49]:

```
samples_mean = np.zeros(10000)
samples_median = np.zeros(10000)
samples_mode = np.zeros(10000)
mu, sigma = 20, 3
for s in range(10000):
    sample = np.random.normal(mu, sigma, 250)  #sample of size 250
    x_bar = sample.mean()                      #calculating sample mean
    samples_mean[s] = x_bar
    samples_median[s] = np.median(sample)
    samples_mode[s] = mode(sample).mode[0]
sns.distplot(samples_mean, hist = True, label = 'Mean_Dist',
             hist_kws={'edgecolor':'black'}).set(xlabel = 'Sample Mean',
                                                 ylabel = 'Density')
sns.distplot(samples_median, hist = True, label = 'Median_Dist',
             hist_kws={'edgecolor':'black'}).set(xlabel = 'Sample Median',
                                                  ylabel = 'Density')
sns.distplot(samples_mode, hist = True, label = 'Mode_Dist',
             hist_kws={'edgecolor':'black'}).set(xlabel = 'Sample Median',
                                                  ylabel = 'Density')
plt.legend()
```

Out[49]: <matplotlib.legend.Legend at 0x21d28f9e8e0>



No conclusive statement form MODE