Lab 8: Normal Distribution and Variance of Sample Means

Welcome to Lab 8!

In today's lab, we will learn about the variance of sample means (https://www.inferentialthinking.com/chapters/14/5/variability-of-the-sample-mean.html) as well as the normal distribution (https://www.inferentialthinking.com/chapters/14/3/SD_and_the_Normal_Curve.html).

In [5]:
```
# Run this cell, but please don't change it.

# These lines import the Numpy and Datascience modules.
import numpy as np
from datascience import *

# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)

# These lines load the tests.
from client.api.notebook import Notebook
ok = Notebook('lab08.ok')
_ = ok.submit()
```

```
=====================================================================
Assignment: Normal Distribution and Variance of Sample Means
OK, version v1.14.20
=====================================================================



Saving notebook... No valid file sources found
Submit... 0.0% complete
Could not submit: Assignment does not exist
Backup... 0.0% complete
Could not backup: Assignment does not exist
```

1. Normal Distributions

When we visualize the distribution of a sample, we are often interested in the mean and the standard deviation of the sample (for the rest of this lab, we will abbreviate “standard deviation” as “SD”). These two summary statistics can give us a bird’s eye view of the distribution - by letting us know where the distribution sits on the number line and how spread out it is, respectively.

We want to check if the data is linearly related, so we should look at the data.

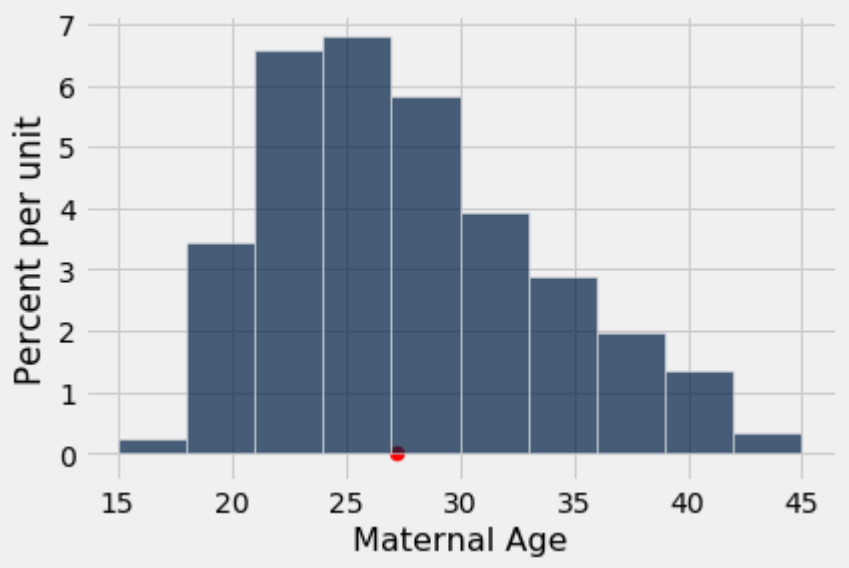
Question 1.1. The next cell loads the table `births` from lecture, which is a large random sample of US births and includes information about mother-child pairs.

Plot the distribution of mother's ages from the table. Don't change the last line, which will plot the mean of the sample on the distribution itself.

```
BEGIN QUESTION
name: q1_1
```

In [6]:
```
births = Table.read_table('baby.csv')
births.hist("Maternal Age") # SOLUTION

# Do not change this line
plt.scatter(np.mean(births.column("Maternal Age")), 0, color='red', s=
50);
```

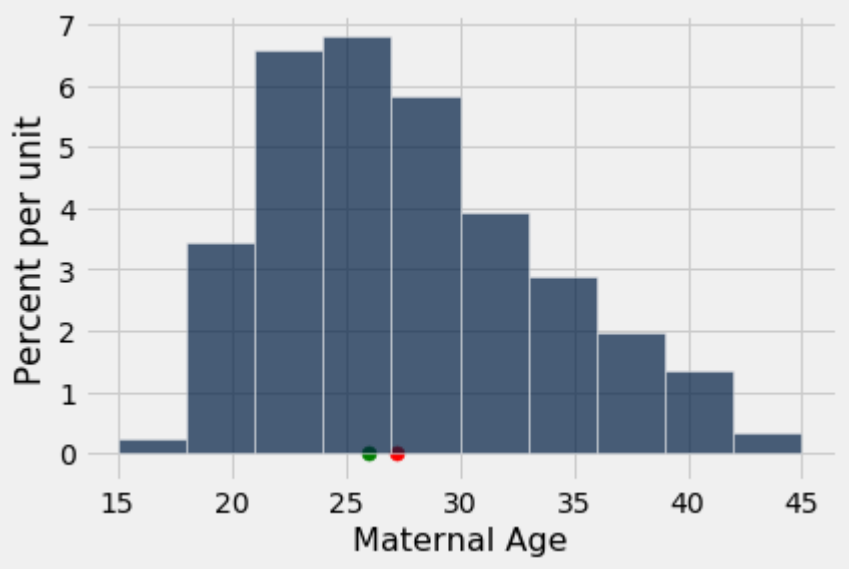


From the plot above, we can see that the mean is the center of gravity or balance point of the distribution. If you cut the distribution out of cardboard, and then placed your finger at the mean, the distribution would perfectly balance on your finger. Since the distribution above is right skewed (which means it has a long right tail), we know that the mean of the distribution is larger than the median, which is the "halfway" point of the data. Conversely, if the distribution had been left skewed, we know the mean would be smaller than the median.

Question 1.2. Run the following cell to compare the mean (red) and median (green) of the distribution of mothers ages.

```
BEGIN QUESTION
name: q1_2
```

In [7]:
```
births.hist("Maternal Age")
plt.scatter(np.mean(births.column("Maternal Age")), 0, color='red', s=
50);
plt.scatter(np.median(births.column("Maternal Age")), 0, color='green'
, s=50);
```



We are also interested in the standard deviation of mother's ages. The SD gives us a sense of how variable mothers' ages are around the average mothers' age. If the SD is large, then the mothers' heights should spread over a large range from the mean. If the SD is small, then the mothers' heights should be tightly clustered around the average mother height.

The SD of an array is defined as the root mean square of deviations (differences) from average.

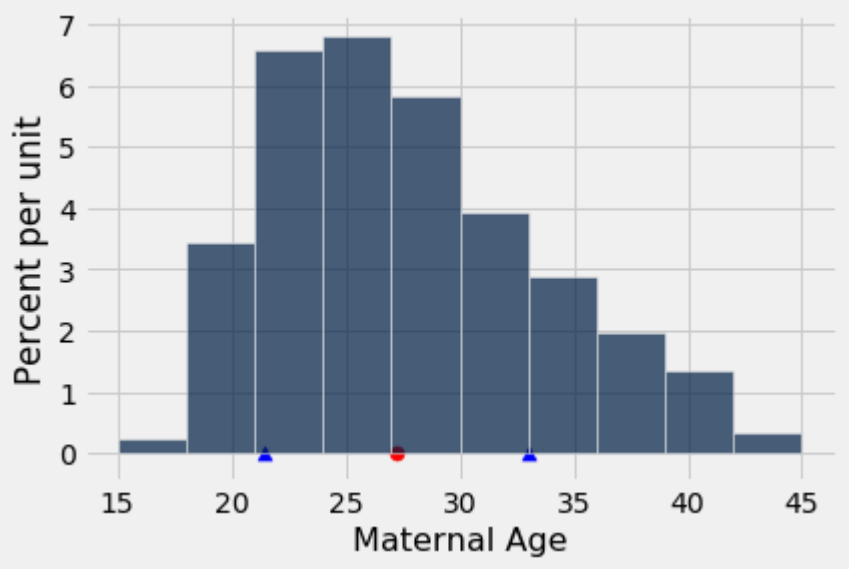
Fun fact! σ (Greek letter sigma) is used to represent the SD and μ (Greek letter mu) is used for the mean.

Question 1.3. Run the cell below to see the width of one SD (blue) from the sample mean (red) plotted on the histogram of maternal ages.

```
BEGIN QUESTION
name: q1_3
```

In [8]:
```
age_mean = np.mean(births.column("Maternal Age")) # SOLUTION
age_sd = np.std(births.column("Maternal Age")) # SOLUTION
births.hist("Maternal Age")

plt.scatter(age_mean, 0, color='red', s=50);
plt.scatter(age_mean+age_sd, 0, marker='^', color='blue', s=50);
plt.scatter(age_mean-age_sd, 0, marker='^', color='blue', s=50);
```



In this histogram, the standard deviation is not easy to identify just by looking at the graph.

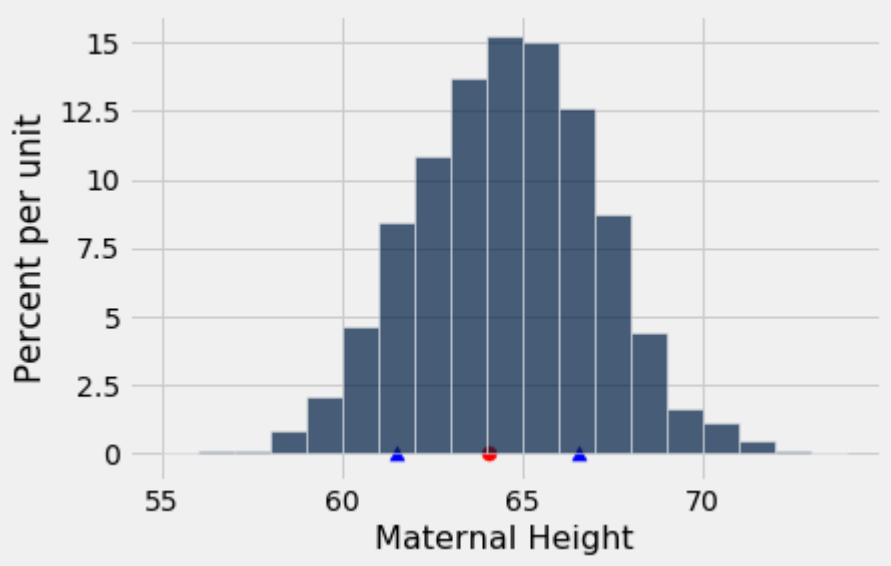
However, the distributions of some variables allow us to easily spot the standard deviation on the plot. For example, if a sample follows a *normal distribution*, the standard deviation is easily spotted at the point of inflection (the point where the curve begins to change the direction of its curvature) of the distribution.

Question 1.4. Fill in the following code to examine the distribution of maternal heights, which is roughly normally distributed. We'll plot the standard deviation on the histogram, as before - notice where one standard deviation (blue) away from the mean (red) falls on the plot.

```
BEGIN QUESTION
name: q1_4
```

In [9]:
```
height_mean = np.mean(births.column("Maternal Height")) # SOLUTION
height_sd = np.std(births.column("Maternal Height")) # SOLUTION
births.hist("Maternal Height", bins=np.arange(55,75,1))

plt.scatter((height_mean), 0, color='red', s=50);
plt.scatter(height_mean+height_sd, 0, marker='^', color='blue', s=50);
plt.scatter(height_mean-height_sd, 0, marker='^', color='blue', s=50);
```



We don’t always know how a variable will be distributed, and making assumptions about whether or not a variable will follow a normal distribution is dangerous. However, the Central Limit Theorem defines one distribution that always follows a normal distribution. The distribution of the *sums* and *means* of many large random samples drawn with replacement from a single distribution (regardless of the distribution’s original shape) will be normally distributed. Remember that the Central Limit Theorem refers to the distribution of a *statistic* calculated from a distribution, not the distribution of the original sample or population. If this is confusing, ask a TA!

The next section will explore distributions of sample means, and you will see how the standard deviation of these distributions depends on sample sizes.

2. Variability of the Sample Mean

By the Central Limit Theorem (https://www.inferentialthinking.com/chapters/14/4/Central_Limit_Theorem.html), the probability distribution of the mean of a large random sample is roughly normal. The bell curve is centered at the population mean. Some of the sample means are higher and some are lower, but the deviations from the population mean are roughly symmetric on either side, as we have seen repeatedly. Formally, probability theory shows that the sample mean is an **unbiased estimate** of the population mean.

In our simulations, we also noticed that the means of larger samples tend to be more tightly clustered around the population mean than means of smaller samples. In this section, we will quantify the variability of the sample mean (https://www.inferentialthinking.com/chapters/14/5/Variability_of_the_Sample_Mean.html) and develop a relation between the variability and the sample size.

Let's take a look at the salaries of employees of the City of San Francisco in 2014. The mean salary reported by the city government was about $75,463.92.

Note: If you get stuck on any part of this lab, please refer to chapter 14 of the textbook (https://www.inferentialthinking.com/chapters/14/Why_the_Mean_Matters.html).

In [11]:
```
salaries = Table.read_table('sf_salaries_2014.csv').select("salary")
salaries
```

Out[11]:

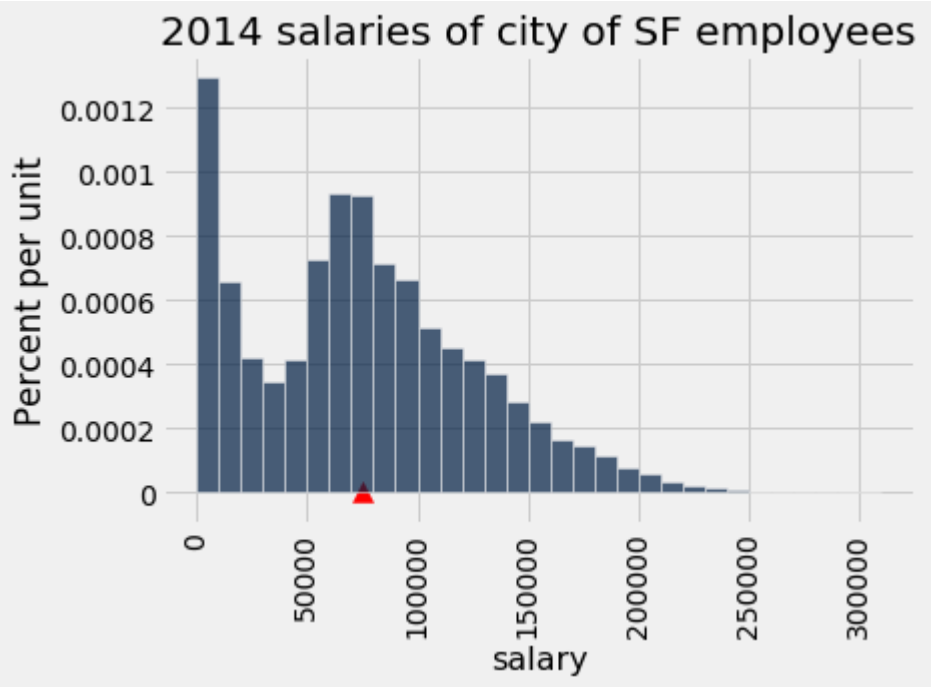
salary
471953
390112
339654
326717
326233
344187
311299
310161
335485
329391

... (38113 rows omitted)

In [12]:
```
salary_mean = np.mean(salaries.column('salary'))
print('Mean salary of San Francisco city employees in 2014: ', salary_mean)
```

```
Mean salary of San Francisco city employees in 2014:  75463.91814023031
```

In [13]:
```
salaries.hist('salary', bins=np.arange(0, 300000+10000*2, 10000))
plt.scatter(salary_mean, 0, marker='^', color='red', s=100);
plt.title('2014 salaries of city of SF employees');
```



Clearly, the population does not follow a normal distribution. Keep that in mind as we progress through these exercises.

Let's take random samples *with replacement* and look at the probability distribution of the sample mean. As usual, we will use simulation to get an empirical approximation to this distribution.

Question 2.1. Define a function `one_sample_mean`. Its arguments should be `table` (the name of a table), `label` (the label of the column containing the variable), and `sample_size` (the number of employees in the sample). It should sample with replacement from the table and return the mean of the `label` column of the sample.

```
BEGIN QUESTION
name: q2_1
```

In [14]:
```
def one_sample_mean(table, label, sample_size):
    new_sample = table.sample(sample_size, with_replacement=True) # SOLUTION
    new_sample_mean = np.mean(new_sample.column(label)) # SOLUTION
    return new_sample_mean # SOLUTION
```

In [15]:
```
# TEST
np.random.seed(8)
one_sample_mean(salaries, 'salary', 100)
```

Out[15]:
```
76699.82860000001
```

Question 2.2. Use `one_sample_mean` to define a function `simulate_sample_mean`. The arguments are the name of the table, the label of the column containing the variable, the sample size, and the number of simulations.

The function should sample with replacement from the table and calculate the mean of each sample. It should save the sample means in an array called `means`. The remaining code in the function displays an empirical histogram of the sample means.

```
BEGIN QUESTION
name: q2_2
```

In [16]:
```
"""Empirical distribution of random sample means"""

def simulate_sample_mean(table, label, sample_size, repetitions):

    means = make_array()

    for i in np.arange(repetitions):
        new_sample_mean = one_sample_mean(table, label, sample_size) #
SOLUTION
        means = np.append(means, new_sample_mean) #SOLUTION

    sample_means = Table().with_column('Sample Means', means)

    # Display empirical histogram and print all relevant quantities -
 don't change this!
    sample_means.hist(bins=20)
    plt.xlabel('Sample Means')
    plt.title('Sample Size ' + str(sample_size))
    print("Sample size: ", sample_size)
    print("Population mean:", np.mean(table.column(label)))
    print("Average of sample means: ", np.mean(means))
    print("Population SD:", np.std(table.column(label)))
    print("SD of sample means:", np.std(means))
    return np.std(means)
```

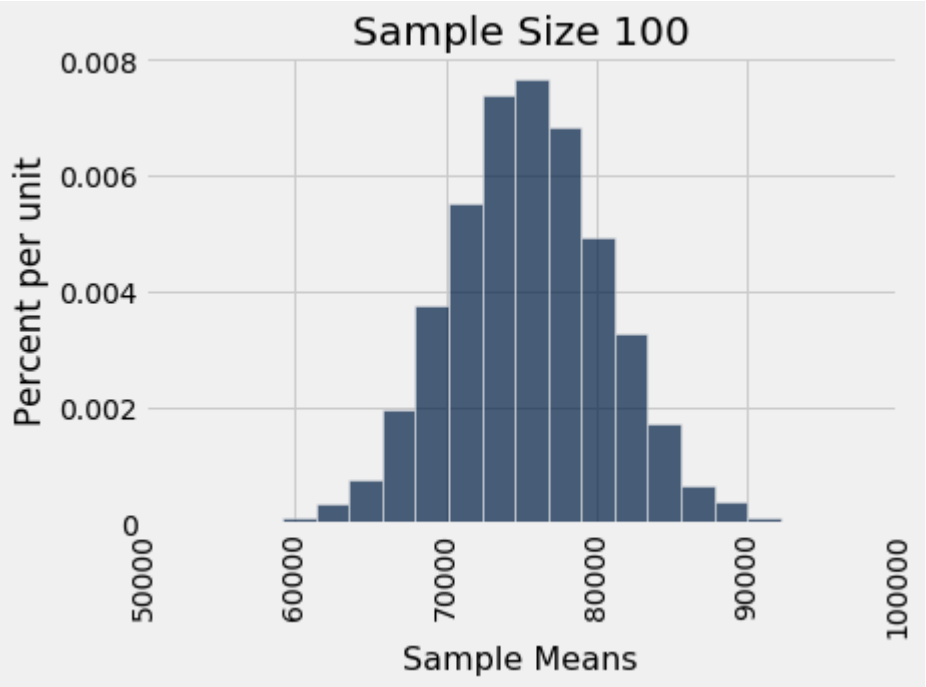
Verify with your neighbor or TA that you've implemented the function above correctly. If you haven't implemented it correctly, the rest of the lab won't work properly, so this step is crucial.

In the following cell, we will create a sample of size 100 from `salaries` and graph it using our new `simulate_sample_mean` function.

Hint: You should see a distribution similar to something we've been talking about. If not, check your function

In [17]:
```
simulate_sample_mean(salaries, 'salary', 100, 10000)
plt.xlim(50000, 100000);
```

```
Sample size:  100
Population mean: 75463.91814023031
Average of sample means:  75465.98348273999
Population SD: 51697.0349864653
SD of sample means: 5126.049251190718
```

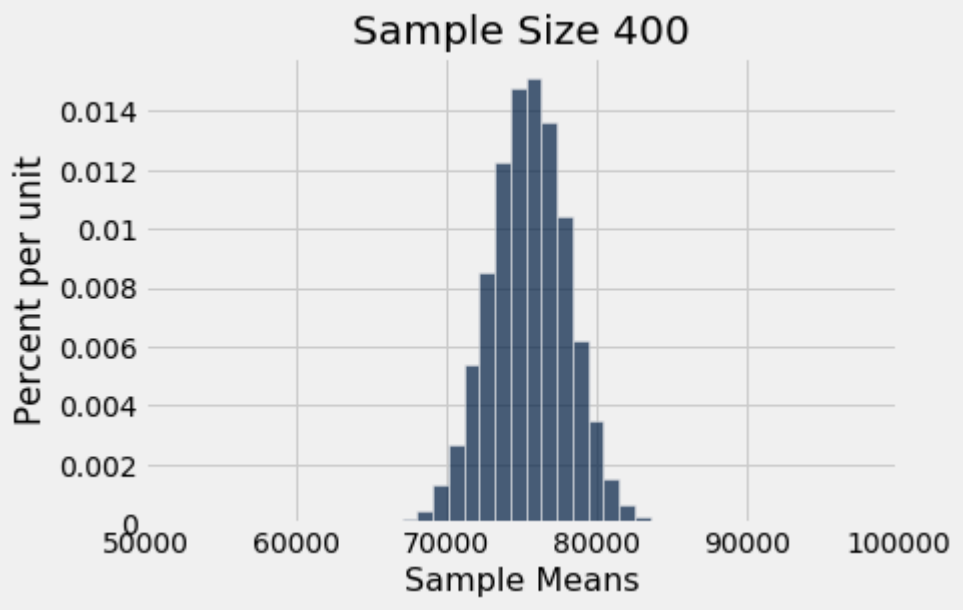


Question 2.3. Simulate two sample means, one for a sample of 400 salaries and one for a sample of 625 salaries. In each case, perform 10,000 repetitions. Don't worry about the `plots.xlim` line – it just makes sure that all of the plots have the same x-axis.

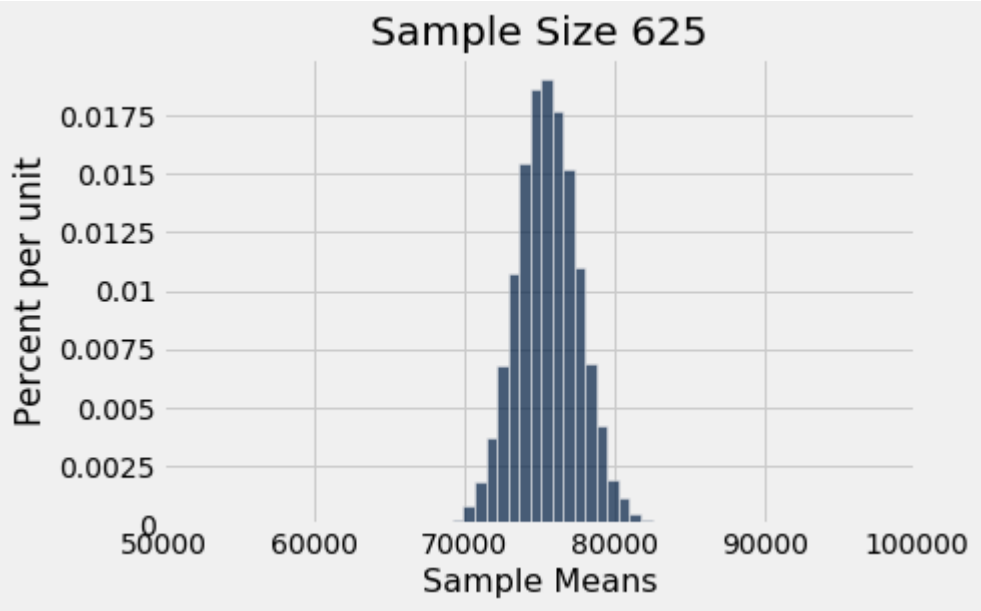
```
BEGIN QUESTION
name: q2_3
```

In [18]:
```
# BEGIN SOLUTION NO PROMPT
simulate_sample_mean(salaries, 'salary', 400, 10000)
plt.xlim(50000, 100000);
plt.show();
print('\n')
simulate_sample_mean(salaries, 'salary', 625, 10000)
plt.xlim(50000, 100000);
plt.show();
# END SOLUTION
""" # BEGIN PROMPT
simulate_sample_mean(..., ..., ..., ...)
plt.xlim(50000, 100000);
plt.show();
print('\n')
simulate_sample_mean(..., ..., ..., ...)
plt.xlim(50000, 100000);
plt.show();
"""; # END PROMPT
```

```
Sample size:  400
Population mean: 75463.91814023031
Average of sample means:  75482.4706507425
Population SD: 51697.0349864653
SD of sample means: 2575.050998699082
```




```
Sample size:  625
Population mean: 75463.91814023031
Average of sample means:  75519.5976711408
Population SD: 51697.0349864653
SD of sample means: 2058.247120966861
```



Question 2.4. Assign `q2_4` to an array of numbers corresponding to true statement(s) about the plots from 2.3.

1. We see the Central Limit Theorem (CLT) in action because the distributions of the sample means are bell-shaped.
2. We see the Law of Averages in action because the distributions of the sample means look like the distribution of the population.
3. One of the conditions for CLT is that we have to draw a small random sample with replacement from the population.
4. One of the conditions for CLT is that we have to draw a large random sample with replacement from the population.
5. One of the conditions for CLT is that the population must be normally distributed.
6. Both plots in 2.3 are roughly centered around the population mean.
7. Both plots in 2.3 are roughly centered around the mean of a particular sample.
8. The distribution of sample means for sample size 625 has less variability than the distribution of sample means for sample size 400.
9. The distribution of sample means for sample size 625 has more variability than the distribution of sample means for sample size 400.

```
BEGIN QUESTION
name: q2_4
```

In [19]:
```
q2_4 = make_array(1, 4, 6, 8) # SOLUTION
```

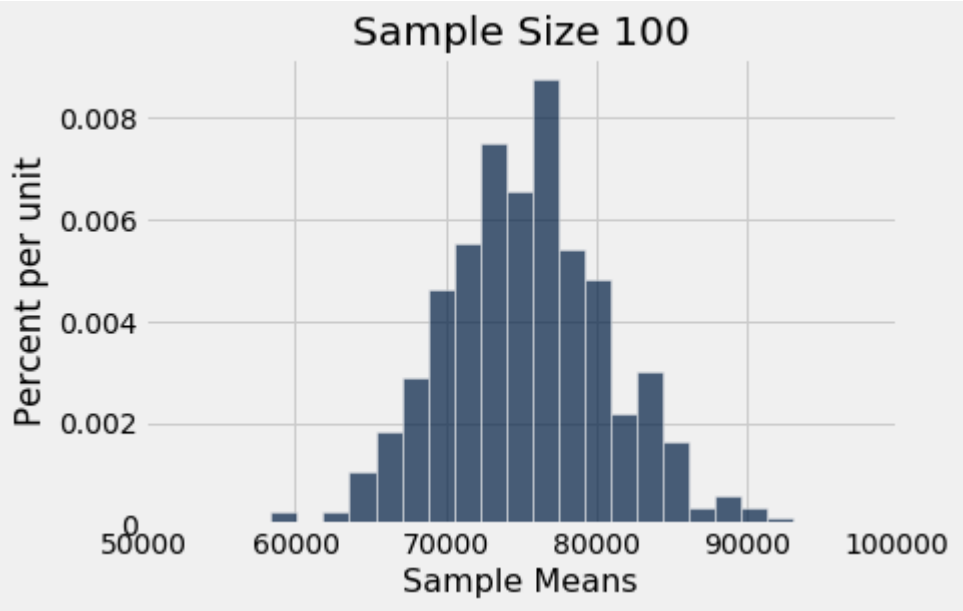
In [20]:
```
# TEST
set(q2_4) == set([1, 4, 6, 8])
```

Out[20]: True

Below, we'll look at what happens when we take an increasing number of resamples of a fixed sample size. Notice what number in the code changes, and what stays the same. How does the distribution of the resampled means change?

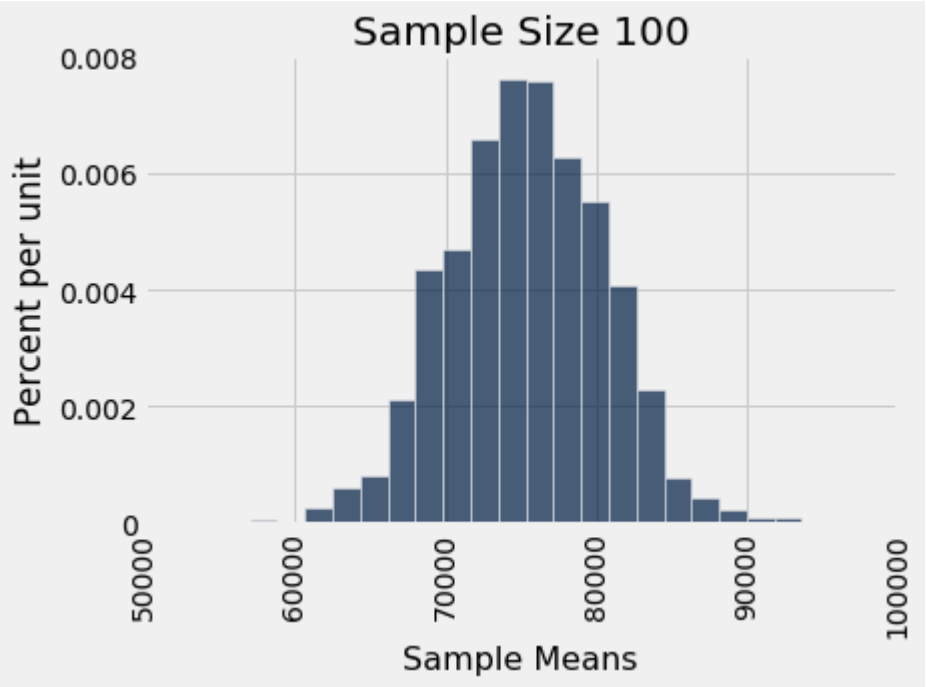
In [21]:
```
simulate_sample_mean(salaries, 'salary', 100, 500)
plt.xlim(50000, 100000);
```

```
Sample size:  100
Population mean: 75463.91814023031
Average of sample means:  75317.2042906
Population SD: 51697.0349864653
SD of sample means: 5402.794055300018
```



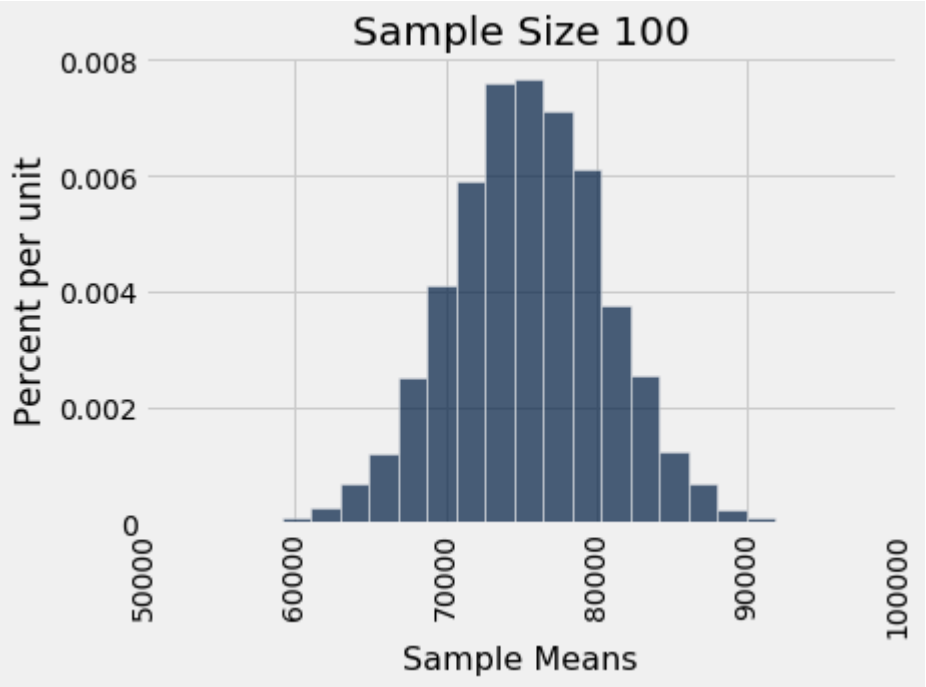
In [22]:
```
simulate_sample_mean(salaries, 'salary', 100, 1000)
plt.xlim(50000, 100000);
```

```
Sample size:  100
Population mean: 75463.91814023031
Average of sample means:  75357.38307030001
Population SD: 51697.0349864653
SD of sample means: 5122.214752913259
```



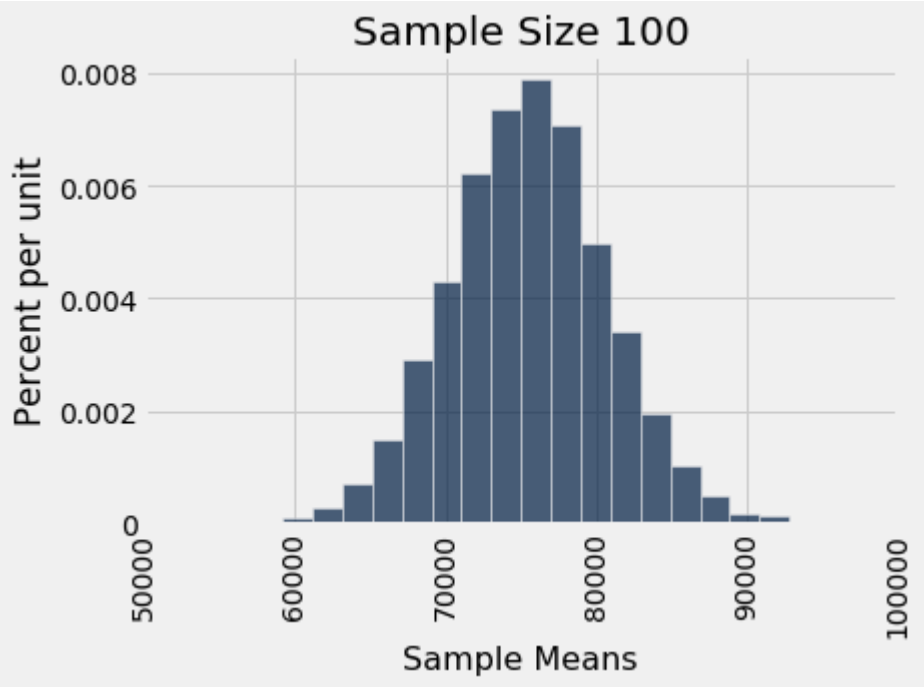
In [23]:
```
simulate_sample_mean(salaries, 'salary', 100, 5000)
plt.xlim(50000, 100000);
```

```
Sample size:  100
Population mean: 75463.91814023031
Average of sample means:  75469.18343674
Population SD: 51697.0349864653
SD of sample means: 5093.39909573918
```



In [24]:
```
simulate_sample_mean(salaries, 'salary', 100, 10000)
plt.xlim(50000, 100000);
```

```
Sample size:  100
Population mean: 75463.91814023031
Average of sample means:  75453.16759068999
Population SD: 51697.0349864653
SD of sample means: 5167.6716961183865
```



What did you notice about the distributions of sample means in the four histograms above? Discuss with your neighbors. If you're unsure of your conclusion, ask your TA.

Question 2.5. Assign the variable `SD_of_sample_means` to the integer corresponding to your answer to the following question:

When I increase the number of resamples that I take, for a fixed sample size, the SD of my sample means will...

1. Increase
2. Decrease
3. Stay about the same
4. Vary widly

```
BEGIN QUESTION
name: q2_5
```

In [25]:
```
SD_of_sample_means = 3 #SOLUTION
```

In [26]:
```
# TEST
SD_of_sample_means == 3
```

Out[26]: True

Question 2.6. Let's think about how the relationships between population SD, sample SD, and SD of sample means change with varying sample size. Which of the following is true? Assign the variable `pop_vs_sample` to an array of integer(s) that correspond to true statement(s).

1. Sample SD gets smaller with increasing sample size.
2. Sample SD gets larger with increasing sample size.
3. Sample SD becomes more consistent with population SD with increasing sample size.
4. SD of sample means gets smaller with increasing sample size.
5. SD of sample means gets larger with increasing sample size.
6. SD of sample means stays the same with increasing sample size.

```
BEGIN QUESTION
name: q2_6
```

In [27]:
```
pop_vs_sample = make_array(3, 4) # SOLUTION
```

In [28]:
```
# TEST
set(pop_vs_sample) == set([3,4])
```

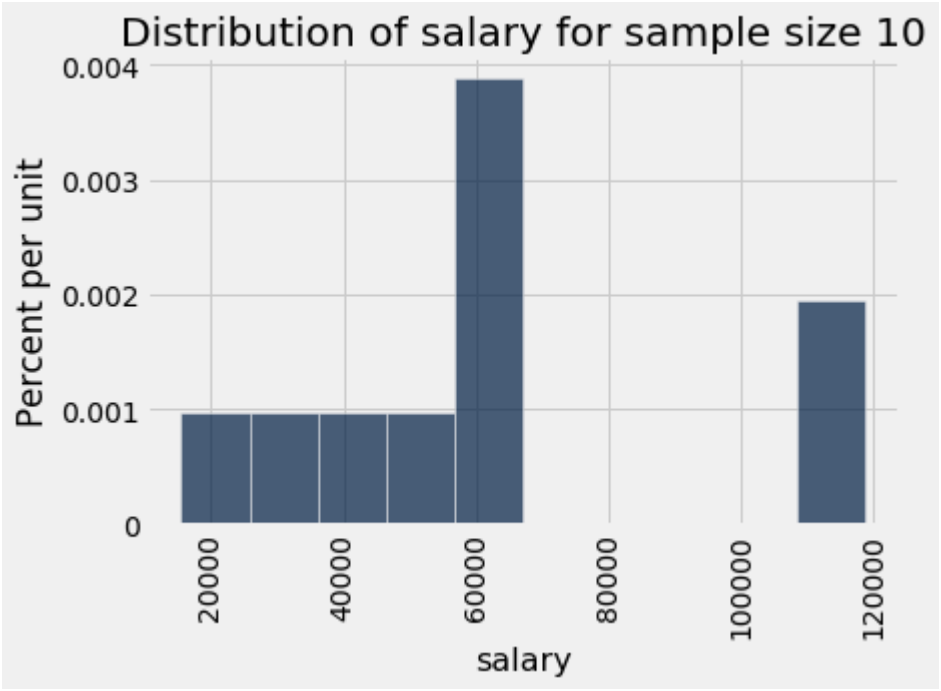
Out[28]: True

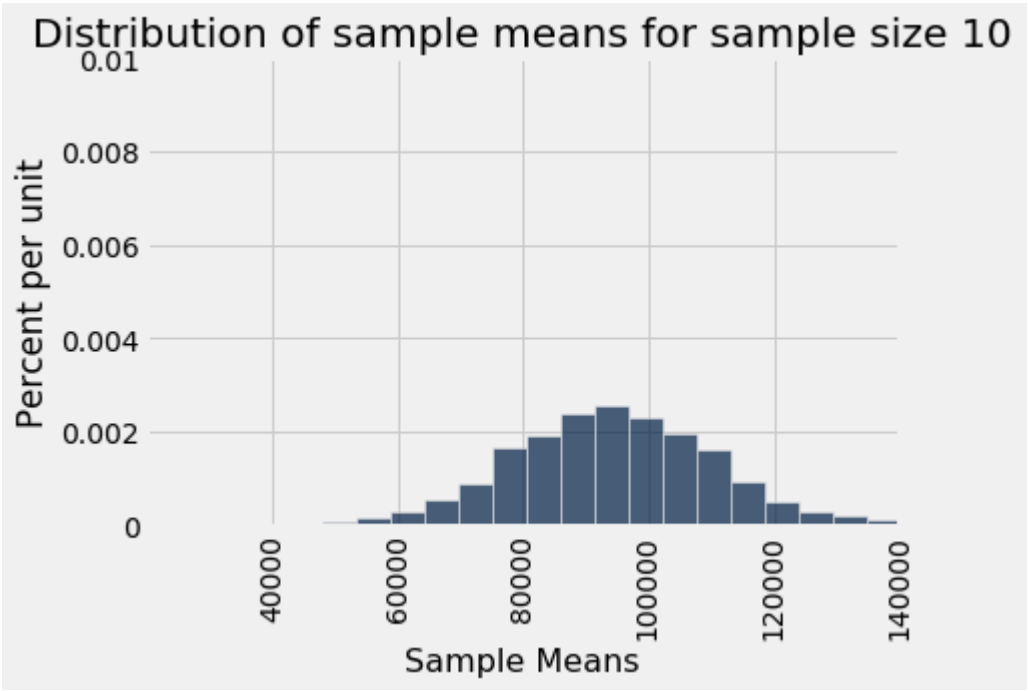
Run the following three cells multiple times and examine how the sample SD and the SD of sample means change with sample size.

The first histogram is of the sample; the second histogram is the distribution of sample means with that particular sample size. Adjust the bins as necessary.

In [29]:
```
sample_10 = salaries.sample(10)
sample_10.hist("salary")
plt.title('Distribution of salary for sample size 10')
print("Sample SD: ", np.std(sample_10.column("salary")))
simulate_sample_mean(salaries, 'salary', 10, 1000)
plt.xlim(5,120000);
plt.ylim(0, .0001);
plt.title('Distribution of sample means for sample size 10');
```

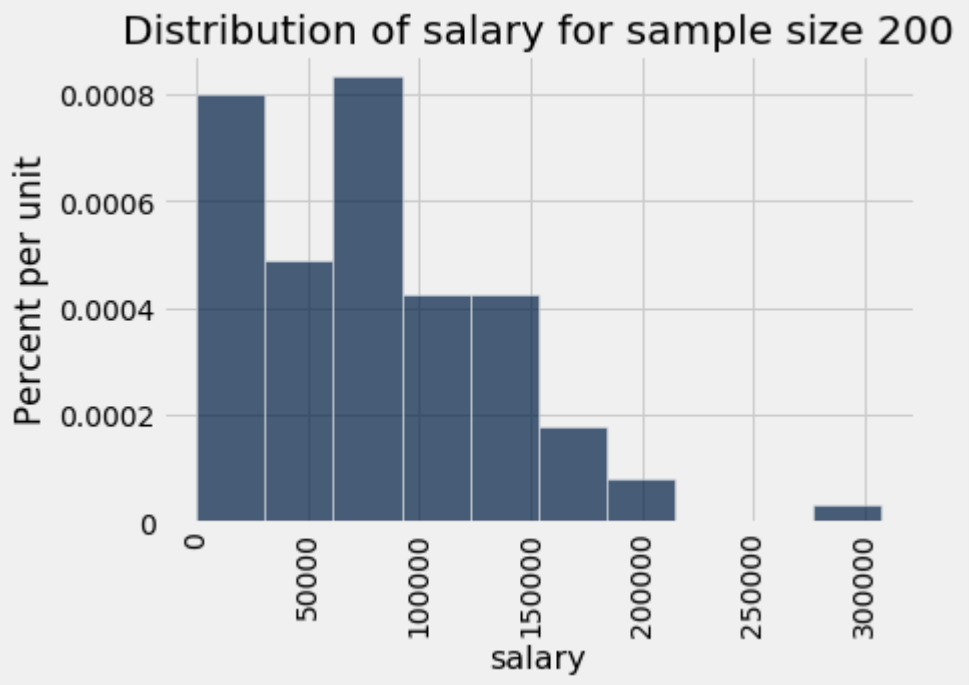
```
Sample SD:  30894.534976741583
Sample size:  10
Population mean: 75463.91814023031
Average of sample means:  74682.91580199999
Population SD: 51697.0349864653
SD of sample means: 16017.633608016884
```

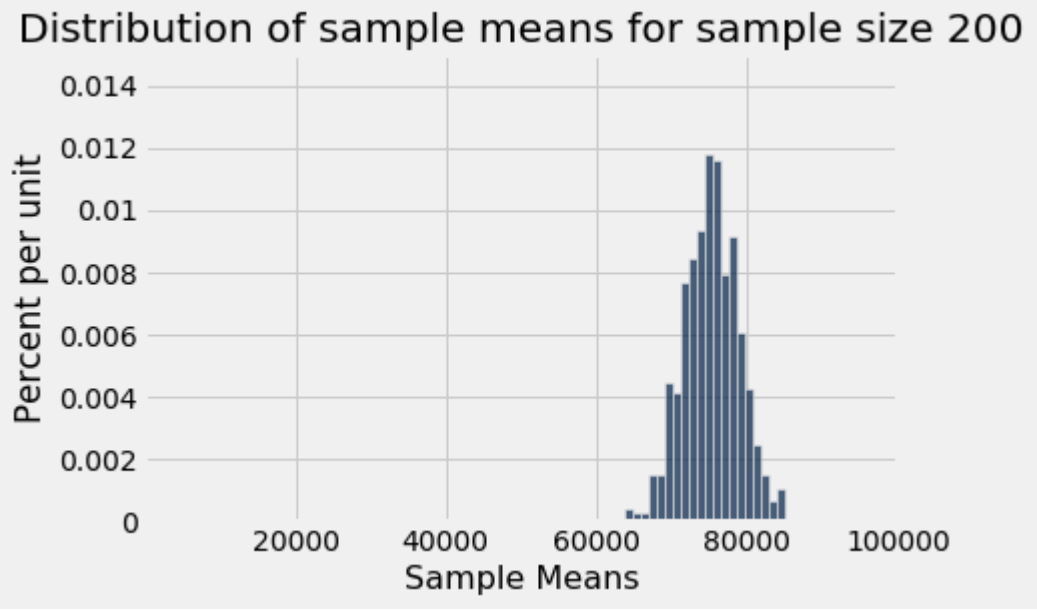




In [30]:
```
sample_200 = salaries.sample(200)
sample_200.hist("salary")
plt.title('Distribution of salary for sample size 200')
print("Sample SD: ", np.std(sample_200.column("salary")))
simulate_sample_mean(salaries, 'salary', 200, 1000)
plt.xlim(5,100000)
plt.ylim(0, .00015);
plt.title('Distribution of sample means for sample size 200');
```

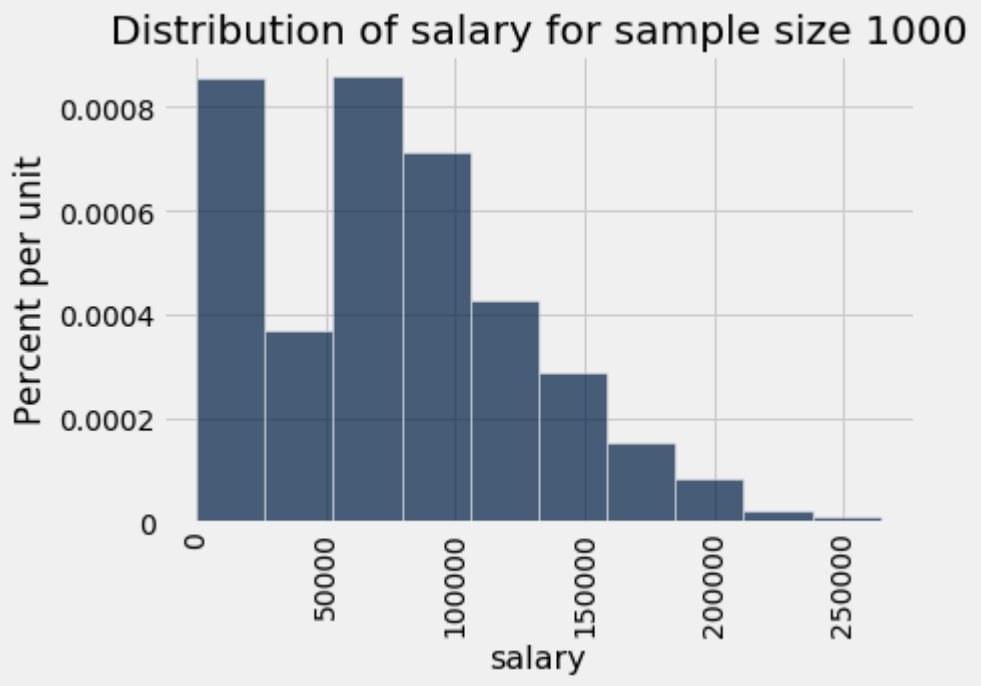
```
Sample SD:  55893.37031987198
Sample size:  200
Population mean: 75463.91814023031
Average of sample means:  75276.06620550001
Population SD: 51697.0349864653
SD of sample means: 3597.2808178253526
```

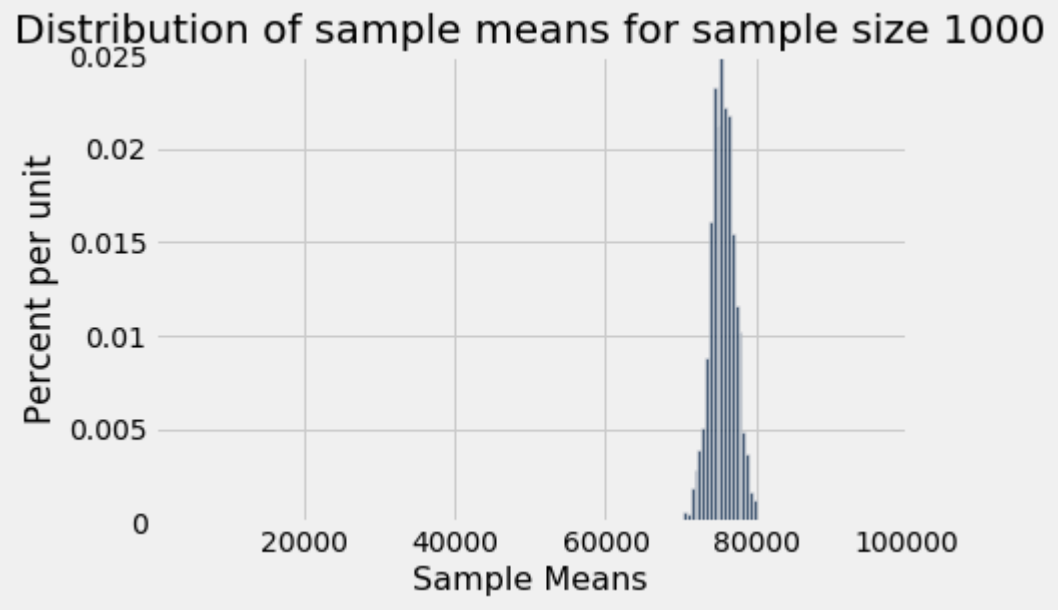




In [31]:
```
sample_1000 = salaries.sample(1000)
sample_1000.hist("salary")
plt.title('Distribution of salary for sample size 1000')
print("Sample SD: ", np.std(sample_1000.column("salary")))
simulate_sample_mean(salaries, 'salary', 1000, 1000)
plt.xlim(5,100000)
plt.ylim(0, .00025);
plt.title('Distribution of sample means for sample size 1000');
```

```
Sample SD:  52155.127103327555
Sample size:  1000
Population mean: 75463.91814023031
Average of sample means:  75505.05874641001
Population SD: 51697.0349864653
SD of sample means: 1627.8423785870941
```

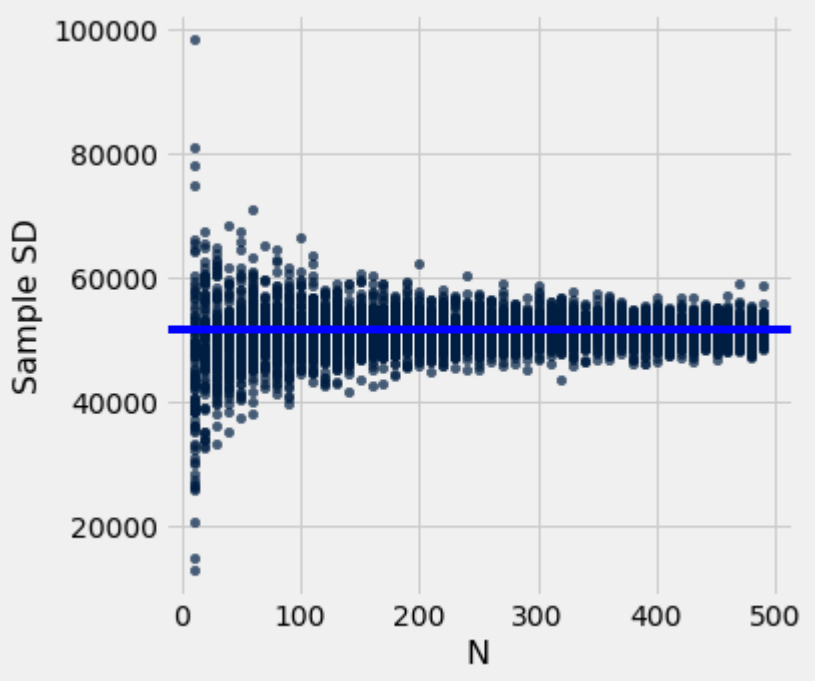




You should notice that the distribution of means gets narrower and spikier, and that the distribution of the sample increasingly looks like the distribution of the population as we get to larger sample sizes.

Let's illustrate these trends. Below, you will see how the sample SD changes with respect to sample size (N). The blue line is the population SD.

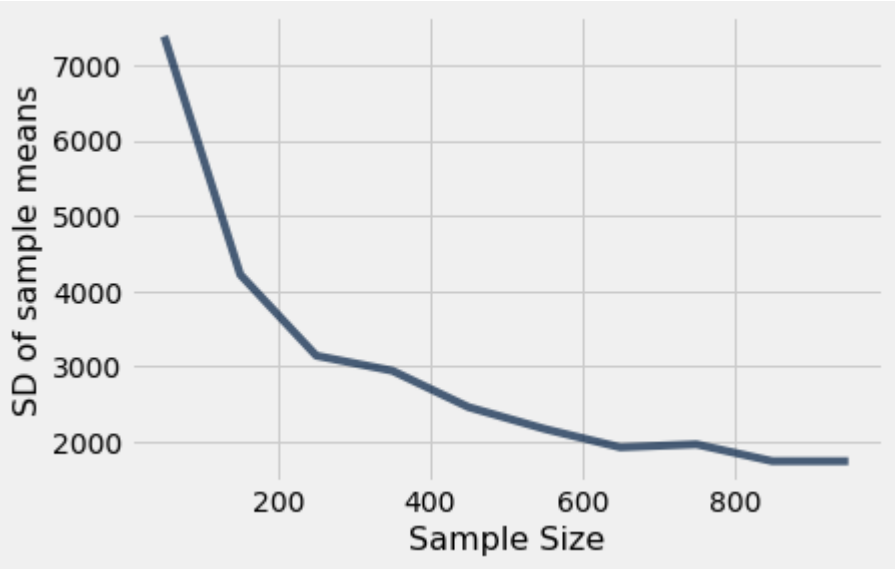
In [32]:
```
# Don't change this cell, just run it!
pop_sd = np.std(salaries.column('salary'))
sample_sds = make_array()
sample_sizes = make_array()
for i in np.arange(10, 500, 10):
    sample_sds = np.append(sample_sds, [np.std(salaries.sample(i).colu
mn("salary")) for d in np.arange(100)])
    sample_sizes = np.append(sample_sizes, np.ones(100) * i)
Table().with_columns("Sample SD", sample_sds, "N", sample_sizes).scatt
er("N", "Sample SD")
matplotlib.pyplot.axhline(y=pop_sd, color='blue', linestyle='-');
```



The next cell shows how the SD of the sample means changes relative to the sample size (N).

In [33]:
```
# Don't change this cell, just run it!
def sample_means(sample_size):
    means = make_array()
    for i in np.arange(1000):
        sample = salaries.sample(sample_size).column('salary')
        means = np.append(means, np.mean(sample))
    return np.std(means)

sample_mean_SDs = make_array()
for i in np.arange(50, 1000, 100):
    sample_mean_SDs = np.append(sample_mean_SDs, sample_means(i))
Table().with_columns("SD of sample means", sample_mean_SDs, "Sample Si
ze", np.arange(50, 1000, 100))\
.plot("Sample Size", "SD of sample means")
```



From these two plots, we can see that the SD of our *sample* approaches the SD of our population as our sample size increases, but the SD of our *sample means* (in other words, the variability of the sample mean) decreases as our sample size increases.

Question 2.7. Is there a relationship between the sample size and the standard deviation of the sample mean? Assign `q2_7` to the number corresponding to the statement that answers this question.

1. The SD of the sample means is inversely proportional to the square root of sample size.
2. The SD of the sample means is directly proportional to the square root of sample size.

```
BEGIN QUESTION
name: q2_7
```

In [34]:
```
q2_7 = 1 # SOLUTION
```

In [35]:
```
# TEST
q2_7 == 1
```

Out[35]:
```
True
```

Throughout this lab, we have been taking many random samples from a population. However, all of these principles hold for bootstrapped resamples from a single sample. If your original sample is relatively large, all of your re-samples will also be relatively large, and so the SD of resampled means will be relatively small.

In order to change the variability of your sample mean, you'd have to change the size of the original sample from which you are taking bootstrapped resamples.

That's it! You've completed Lab 8. There weren't many tests, but there were a lot of points at which you should've stopped and understood exactly what was going on. Consult the textbook or ask your TA if you have any other questions!

Be sure to

- **run all the tests** (the next cell has a shortcut for that),
- **Save and Checkpoint** from the `File` menu,
- **run the last cell to submit your work**,
- and ask one of the staff members to check you off.

In [36]:
```
# For your convenience, you can run this cell to run all the tests at
 once!
import os
print("Running all tests...")
_ = [ok.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q'
)]
print("Finished running all tests.")
```

```
Running all tests...
Finished running all tests.
```

In [37]:
```
# Run this cell to submit your work *after* you have passed all of the
test cells.
# It's ok to run this cell multiple times. Only your final submission
 will be scored.

_ = ok.submit()
```

```
Saving notebook... No valid file sources found
Submit... 0.0% complete
Could not submit: Assignment does not exist
Backup... 0.0% complete
Could not backup: Assignment does not exist
```

In [ ]:
```

```