In [3]:
```
from datascience import *
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
import numpy as np
```

Lecture 23

Percentiles

In [4]:
```
x = make_array(43, 20, 51, 7, 28, 34)
y = np.sort(x)
y
```

Out[4]:
```
array([ 7, 20, 28, 34, 43, 51], dtype=int64)
```

In [5]:
```
0.55 * 6
```

Out[5]:
```
3.3000000000000003
```

In [6]:
```
percentile(55, x)
```

Out[6]:
```
34
```

Bootstrap

In [7]:
```
sf = Table.read_table('san_francisco_2015.csv')
```

In [8]:
```
sf.where('Job', 'Mayor')
```

Out[8]:

Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	Union	Job Family Code	Job Family	Job Code	Job	Employee Identifier	Salarie

Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	Union	Job Family Code	Job Family	Job Code	Job	Employee Identifier	Salarie
Calendar	2015	6	General Administration & Finance	MYR	Mayor	556	Elected Officials	1100	Administrative & Mgmt (Unrep)	1190	Mayor	22433	28896

In [11]:
```
10 * 20 * 52   # hours threshold for part time
```

Out[11]:
```
10400
```

In [12]:
```
sf = sf.where('Total Compensation', are.above(10000))
```

In [13]:
```
sf.num_rows
```

Out[13]:
```
38217
```

In [14]:
```
percentile(50, sf.column('Total Compensation'))
```

Out[14]:
```
107219.48
```

In [15]:
```
our_sample = sf.sample(300, with_replacement=False)
our_sample
```

Out[15]:

Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	Union	Job Family Code	Job Family	Job Code	Job	Emplo Ident
Calendar	2015	4	Community Health	DPH	Public Health	250	SEIU - Health Workers, Local 1021	2300	Nursing	2303	Patient Care Assistant	16
Calendar	2015	2	Public Works, Transportation & Commerce	AIR	Airport Commission	4	Painters, Local 1176	7300	Journeyman Trade	7346	Painter	7

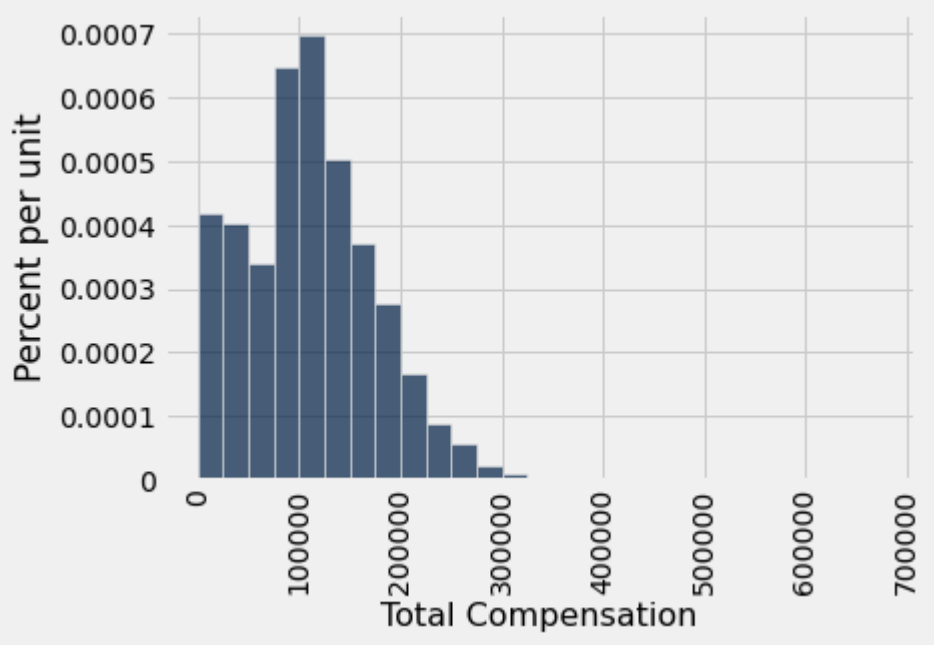
Year Type	Year	Organization Group Code	Organization Group	Department Code	Department	Union Code	Union	Job Family Code	Job Family	Job Code	Job	Emplc Ident
Calendar	2015	1	Public Protection	POL	Police	911	Police Officers' Association	Q000	Police Services	Q002	Police Officer	28
Calendar	2015	6	General Administration & Finance	ADM	General Services Agency - City Admin	790	SEIU - Miscellaneous, Local 1021	1300	Pub Relations & Spec Assts	1324	Customer Service Agent	36
Calendar	2015	2	Public Works, Transportation & Commerce	MTA	Municipal Transportation Agency	253	Transport Workers - Transit Operators, Local 250-A	9100	Street Transit	9163	Transit Operator	50
Calendar	2015	1	Public Protection	FIR	Fire Department	798	Firefighters - Miscellaneous, Local 798	H000	Fire Services	H002	Firefighter	40
Calendar	2015	6	General Administration & Finance	ADM	General Services Agency - City Admin	130	Automotive Machinists, Local 1414	7300	Journeyman Trade	7313	Automotive Machinist	9
Calendar	2015	4	Community Health	DPH	Public Health	856	Teamsters - Miscellaneous, Local 856	2400	Lab, Pharmacy & Med Techs	2462	Microbiologist	25
Calendar	2015	2	Public Works, Transportation & Commerce	MTA	Municipal Transportation Agency	4	Painters, Local 1176	7300	Journeyman Trade	7346	Painter	33
Calendar	2015	2	Public Works, Transportation & Commerce	MTA	Municipal Transportation Agency	21	Prof & Tech Engineers - Miscellaneous, Local 21	1300	Pub Relations & Spec Assts	1314	Public Relations Officer	22

... (290 rows omitted)

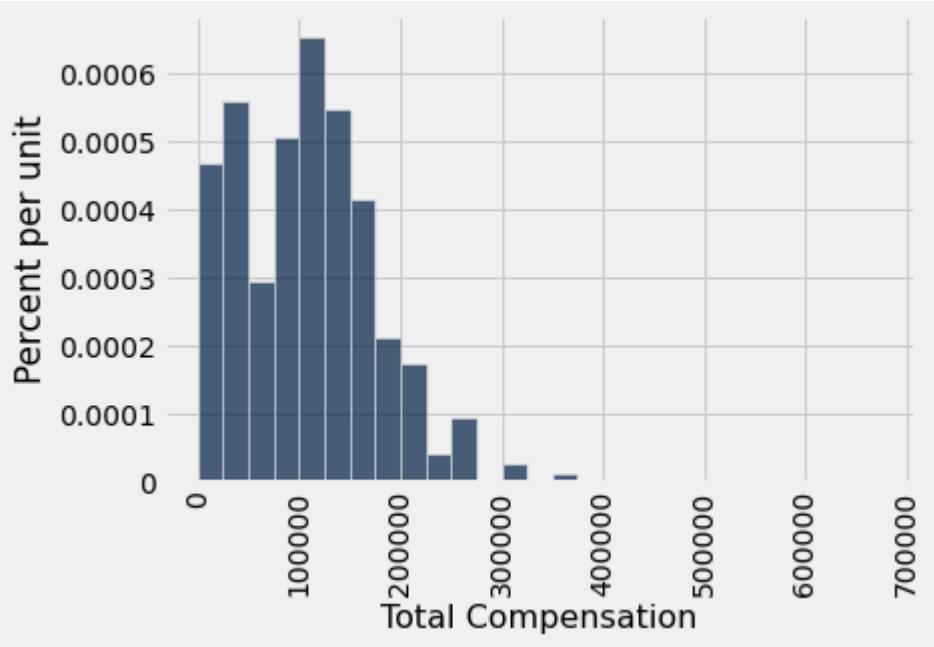
In [16]:
```
percentile(50, our_sample.column('Total Compensation'))
```

Out[16]: 108836.88

In [19]:
```
sf_bins = np.arange(0, 700000, 25000)
sf.hist('Total Compensation', bins=sf_bins)
```

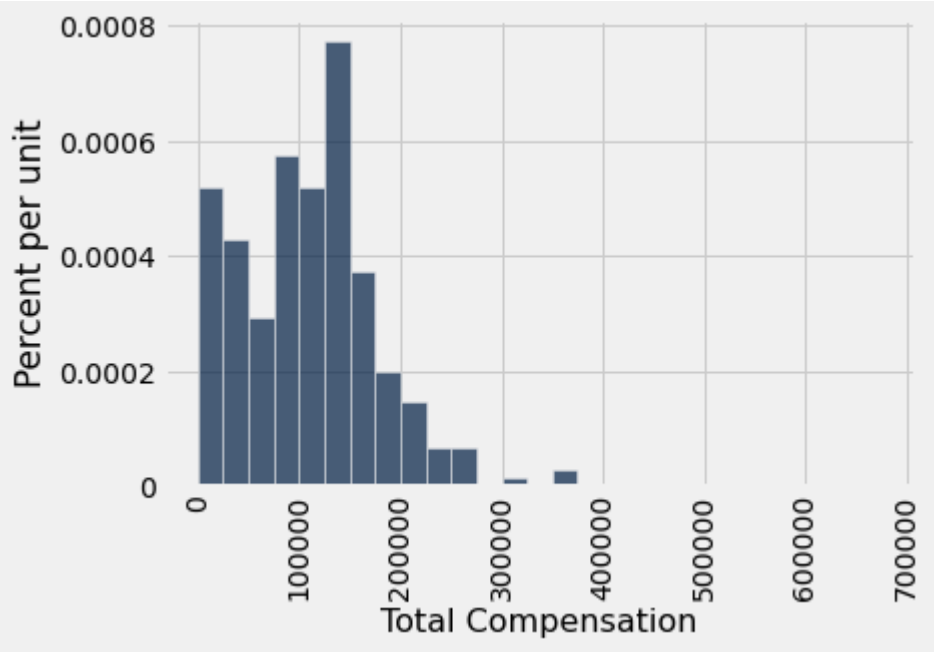


In [20]:
```
our_sample.hist('Total Compensation', bins=sf_bins)
```



In [21]:
```
resample1 = our_sample.sample(300, with_replacement=True)
```

In [22]:
```
resample1.hist('Total Compensation', bins=sf_bins)
```



In [23]:
```
resample2 = our_sample.sample()
```

In [24]:
```
percentile(50, resample1.column('Total Compensation'))
```

Out[24]: 111566.03

In [25]:
```
percentile(50, resample2.column('Total Compensation'))
```

Out[25]: 111491.31

In [26]:
```
medians = make_array()

for i in np.arange(1000):
    resampled = our_sample.sample()
    median = percentile(50, resampled.column('Total Compensation'))
    medians = np.append(medians, median)
```

In [27]:
```
def bootstrap_median(original_sample, label, replications):
```

```
    """Simulate sample median:
    original_sample: table containing the original sample
    label: label of column containing the variable
    replications: number of bootstrap samples
    Returns array of bootstrap sample medians
    """
    medians = make_array()
    for i in np.arange(replications):
        bootstrap_sample = original_sample.sample()
        resampled_median = percentile(50, bootstrap_sample.column(label))
        medians = np.append(medians, resampled_median)

    return medians
```

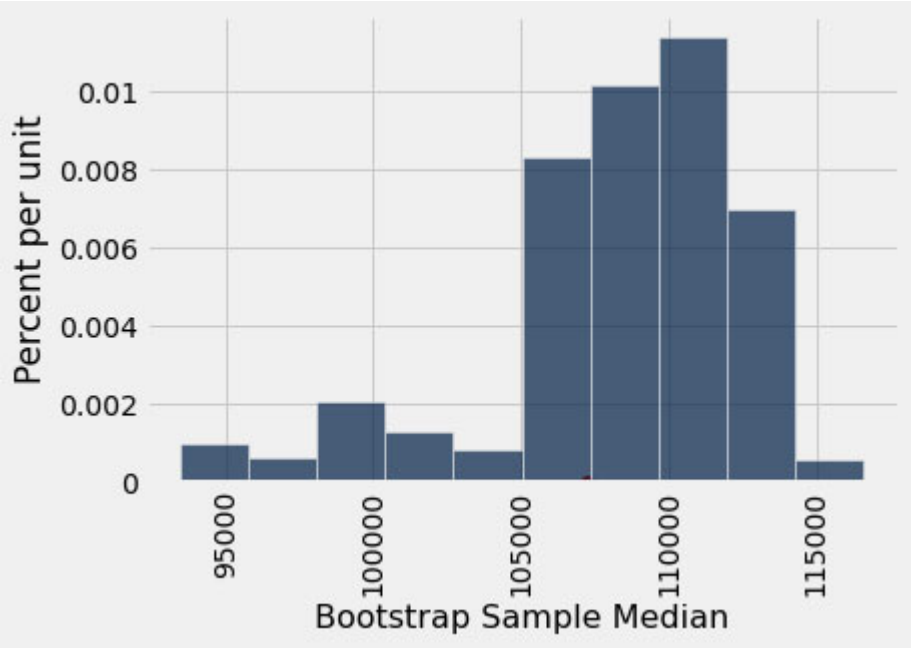
In [28]:
```
bstrap_medians = bootstrap_median(our_sample, 'Total Compensation', 1000)
```

In [29]:
```
pop_median = percentile(50, sf.column('Total Compensation'))
```

In [30]:
```
resampled_medians = Table().with_column(
    'Bootstrap Sample Median', bstrap_medians)

resampled_medians.hist()

plots.scatter(pop_median, 0, color='red', s=40);
```



In [31]:
```
left = percentile(2.5, bstrap_medians)
left
```

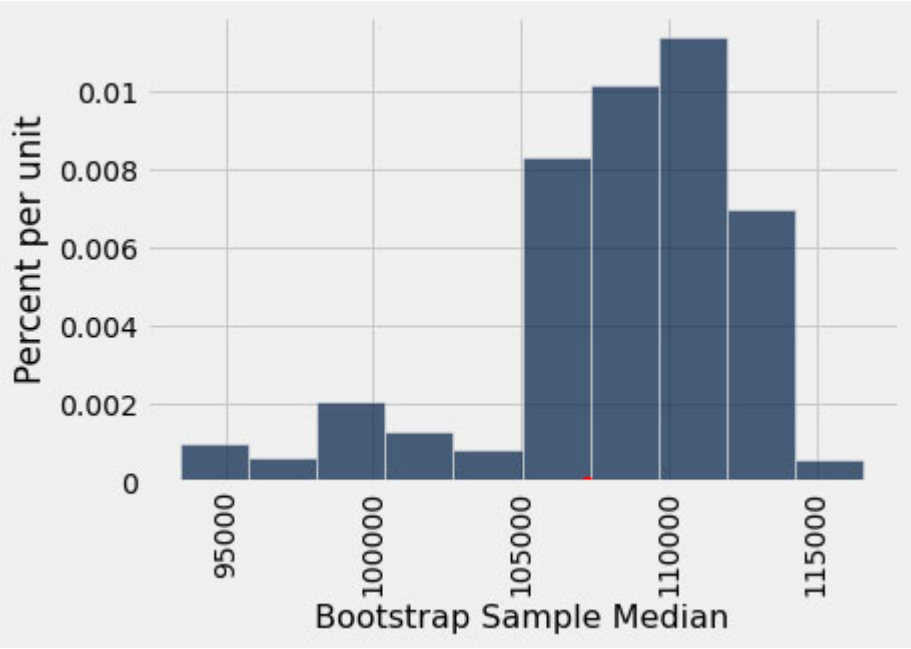
Out[31]: 96183.3

In [32]:
```
right = percentile(97.5, bstrap_medians)
right
```

Out[32]: 113923.18

In [33]:
```
resampled_medians.hist()

plots.plot([left, right], [0, 0], color='yellow', lw=3, zorder=1)
plots.scatter(pop_median, 0, color='red', s=30, zorder=2);
```



In [34]:
```
confidence_interval = make_array(left, right)
confidence_interval
```

Out[34]:
```
array([ 96183.3 , 113923.18])
```

In [35]:
```
# THE BIG SIMULATION: This one takes a long time.

# Generate the endpoints of 50 intervals

left_ends = make_array()
right_ends = make_array()

for i in np.arange(50):
    first_sample = sf.sample(300, with_replacement=False)
    medians = bootstrap_median(first_sample, 'Total Compensation', 2000)
    left_ends = np.append(left_ends, percentile(2.5, medians))
    right_ends = np.append(right_ends, percentile(97.5, medians))
```

In [36]:
```
intervals = Table().with_columns(
    'Left', left_ends,
```

```
    'Right', right_ends
)
```

In [37]:
```
intervals
```

Out[37]:

Left	Right
97710.2	110681
99459	114328
102244	117930
98795	113998
94261.3	110720
95743.9	114497
96973.1	119535
98635.4	113823
96825.8	113068
93830.4	113958

... (40 rows omitted)

In [38]:
```
good = intervals.where('Left', are.below(pop_median)).where('Right', are.above(pop_median)).num_rows
```

In [39]:
```
good / 50
```

Out[39]:
```
0.96
```

In [ ]:
```

```