In [1]:
```
import pandas as pd

import nsfg

import thinkstats2

import thinkplot

import matplotlib.pyplot as plt
```

Final Project Bullet 1

A minimum of 5 variables in your dataset used during your analysis (for help with selecting, the author made his selection on page 6 of your book). Consider what you think could have an impact on your question – remember this is never perfect, so don't be worried if you miss one (Chapter 1).

In [2]:
```
data = pd.read_csv('C:/Users/bonertzb/OneDrive - Tyson Online/Desktop/DataScience 510/
```

In [3]:
```
df = pd.DataFrame(data)
```

In [4]:
```
df.head()
```

Out[4]:

	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID
0	Adinolfi, Wilson K	10026	0	0	1	1	5	4
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3	3
2	Akinkuolie, Sarah	10196	1	1	0	5	5	3
3	Alagbe,Trina	10088	1	1	0	1	5	3
4	Anderson, Carol	10069	0	2	0	5	5	3

5 rows × 36 columns

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Employee_Name', 'EmpID', 'MarriedID', 'MaritalStatusID', 'GenderID',
       'EmpStatusID', 'DeptID', 'PerfScoreID', 'FromDiversityJobFairID',
       'Salary', 'Termd', 'PositionID', 'Position', 'State', 'Zip', 'DOB',
       'Sex', 'MaritalDesc', 'CitizenDesc', 'HispanicLatino', 'RaceDesc',
       'DateofHire', 'DateofTermination', 'TermReason', 'EmploymentStatus',
       'Department', 'ManagerName', 'ManagerID', 'RecruitmentSource',
       'PerformanceScore', 'EngagementSurvey', 'EmpSatisfaction',
       'SpecialProjectsCount', 'LastPerformanceReview_Date', 'DaysLateLast30',
       'Absences'],
      dtype='object')
```

Final Project Bullet 2

Describe what the 5 variables mean in the dataset (Chapter 1).

Variable 1 - EmploymentStatus is a description/category of the person's employment status. This comes important to evaluate if the team member is still employed at the company Variable 2 - Engagement Survey are they results from the last engagement survey. This lists the results of the engagement survey and may provide insight to how engaged team members are with their job responsibilities. This might lead into how well team members are performing their jobs or how well they do their jobs. Variable 3 - Employee Satisfaction is a basic satisfaction score between 1 and 5, as reported on a recent employee satisfaction survey which can provide detail on how well employees like their jobs. This can be used to determine what other attributes or varibales are more or less satisified with their employment. Variable 4 - Salary the person's yearly salary. $ U.S. Dollars. This can be used to compare differences in pay between other attributes or variables and potentially what variables influence a team members salary (absences, department, sex, marital status, etc) Varibale 5 - Absences is the number of times the employee was absent from work. We can use this to compare what other variables have an impact on employee absences, how their pay is influence, if they are more engaged or satisfied with their work, etc.

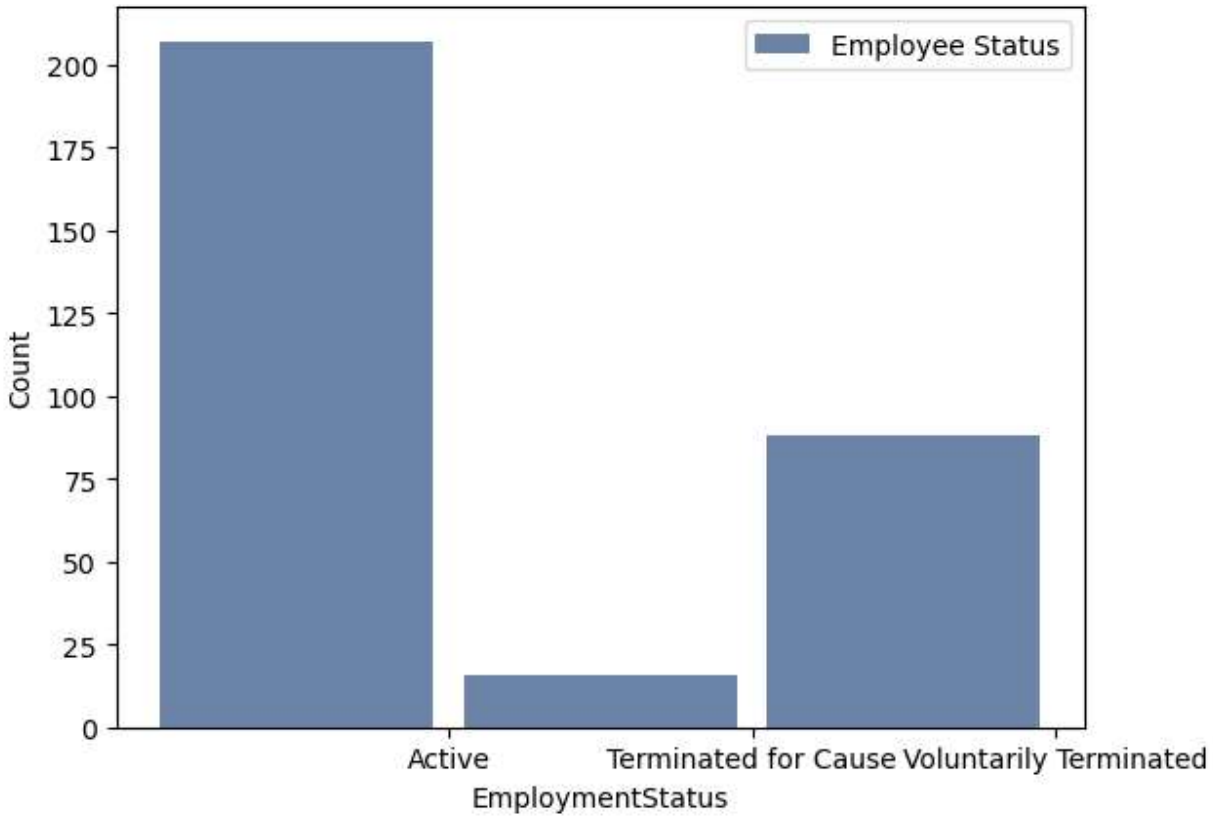
Final Project Bullet 3

Include a histogram of each of the 5 variables – in your summary and analysis, identify any outliers and explain the reasoning for them being outliers and how you believe they should be handled (Chapter 2).

Final Project Bullet 4

Include the other descriptive characteristics about the variables: Mean, Mode, Spread, and Tails (Chapter 2).

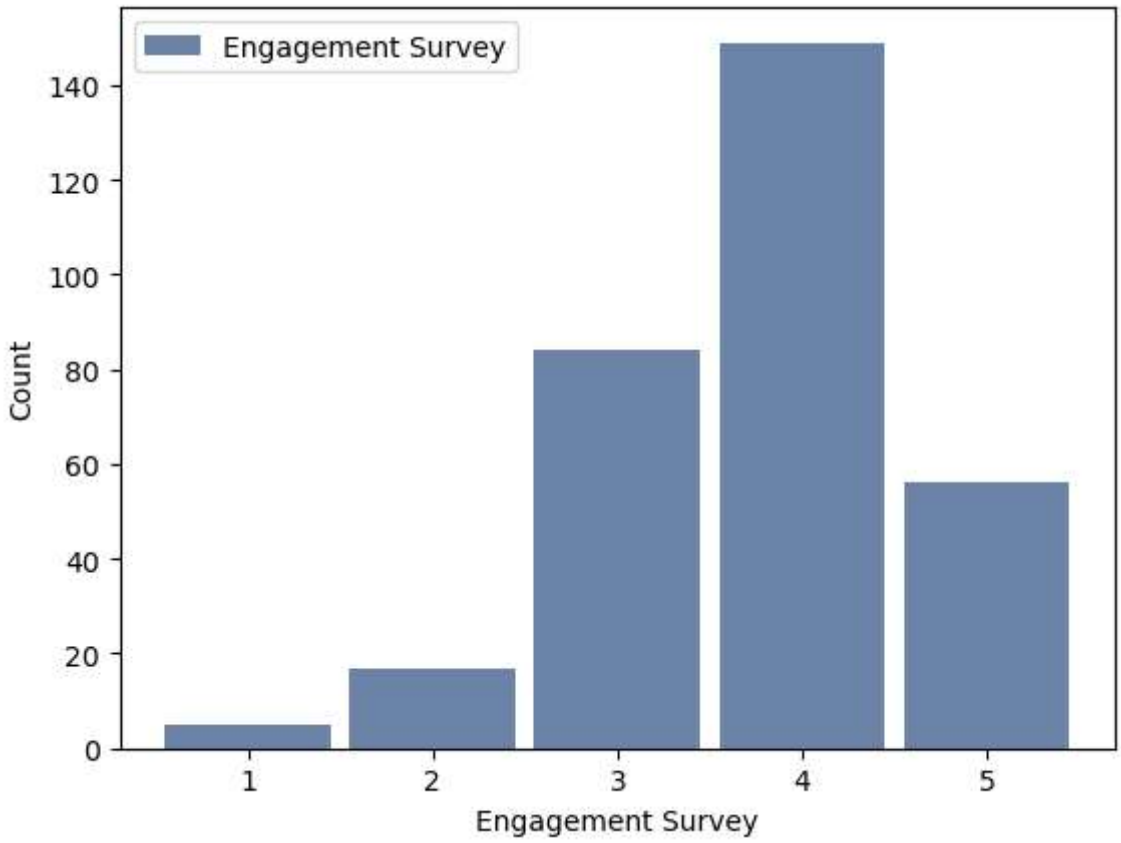
In [6]:
```
#Variale 1 EmploymentStatus, shows the distribution of the employees in the data set t
#and those that were terminated or voluntarily left the company. The mode of the datas
#This variable is of booleen value, so there is no mean but the benefit of this datase
#are still actively contributing information / data to the data set. Furthermore, corr
#the satisfaction rates, engagement surveys, absences, and salary differences between
#terminated, and employees that have voluntarily left the company.

hist = thinkstats2.Hist(df.EmploymentStatus, label='Employee Status')
thinkplot.Hist(hist)
thinkplot.Config(xlabel='EmploymentStatus', ylabel='Count')
```



In [7]:
```
import numpy as np
eng_sur = np.floor(df.EngagementSurvey)

mean_es = df.EngagementSurvey.mean()
max_es = df.EngagementSurvey.max()
min_es = df.EngagementSurvey.min()

mean_es, max_es, min_es
```

Out[7]:
```
(4.11, 5.0, 1.12)
```

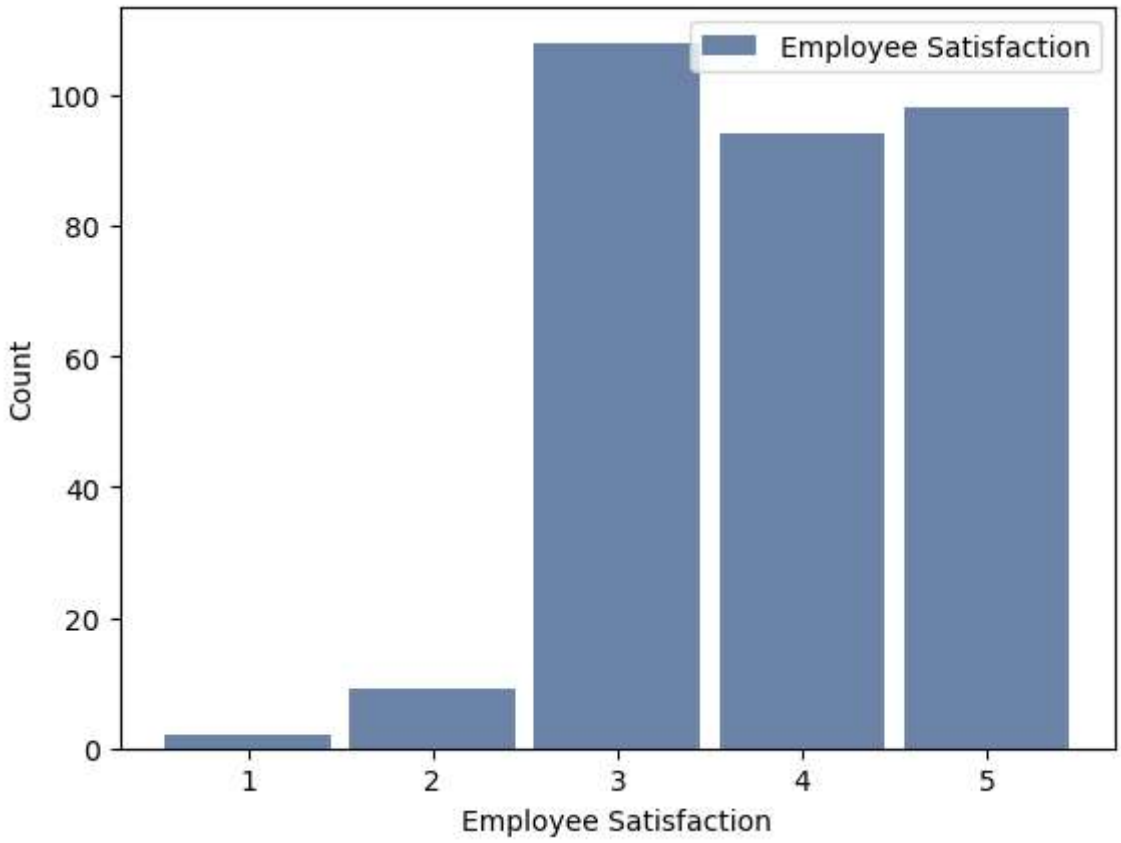
In [8]:
```
#Engagement survey shows the distribution of engagement survey results. Engagement sur
#peaking around 4 with a small skew to the bottom side / left. The mean is 4.11 which
#in the engagement survey were above satisfactory. The engagement survey results asses
#the survey with the max value at 5.0 (assuming a perfect score) and min value of 1.12
#however it highlights the participationof the workers.

hist = thinkstats2.Hist(eng_sur, label='Engagement Survey')
thinkplot.Hist(hist)
thinkplot.Config(xlabel='Engagement Survey', ylabel='Count')
```



In [9]:
```
mean_esat = df.EmpSatisfaction.mean()
max_esat = df.EmpSatisfaction.max()
min_esat = df.EmpSatisfaction.min()

mean_esat, max_esat, min_esat
```

Out[9]:
```
(3.8906752411575565, 5, 1)
```

In [10]:
```
#Employee satisifation is a tool to provide feedback for employers to assess their tea
#The grade scale of the satisfaction survey is from 1 - 5 with a 1 being completely ur
#This data shows that most people are neutral to mostly satisfied with neutral being t
#variable is 3.89 which indicates most of the team members tip right on the edge of ne
#position with a few team members that are not satisfied. This histogram is not evenly
#the left, further supporting the mean and conclusion mentioned earlier.

hist = thinkstats2.Hist(df.EmpSatisfaction, label='Employee Satisfaction')
thinkplot.Hist(hist)
thinkplot.Config(xlabel='Employee Satisfaction', ylabel='Count')
```

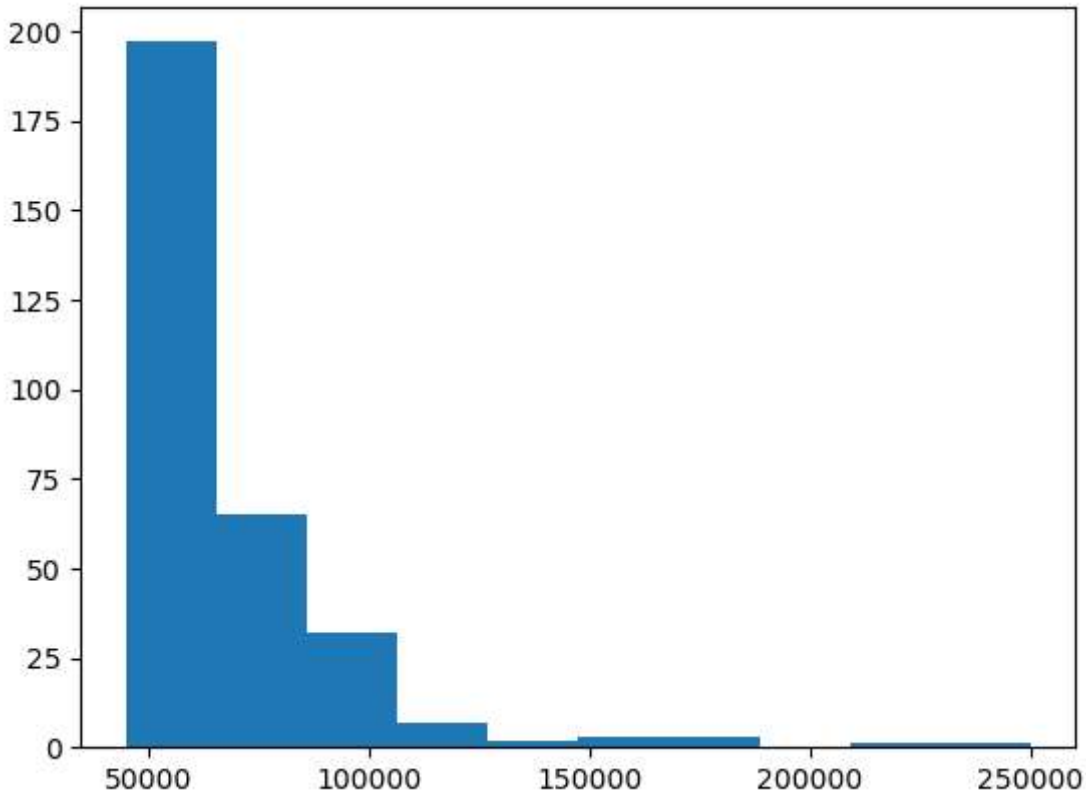


In [11]:
```
mean_sal = df.Salary.mean()
max_sal = df.Salary.max()
min_sal = df.Salary.min()

mean_sal, max_sal, min_sal
```

Out[11]:
```
(69020.6848874598, 250000, 45046)
```

In [12]:
```
#Plot a histogram of the team members salary. The distribution shows the vast majority
#pay scale around the $50,000 range and a few team members in the upper end. The spread
#$45000 - 250000. The mean value of the variable is just shy of $70,000 which is a good
#can be misleading with the distribution skewed with outlier values at the extreme top
#by the mode with stands around the $50,000 dollar mark.
#It will be useful to see how department, team member satisfaction, engagement, absenc

plt.hist(df['Salary'])
```

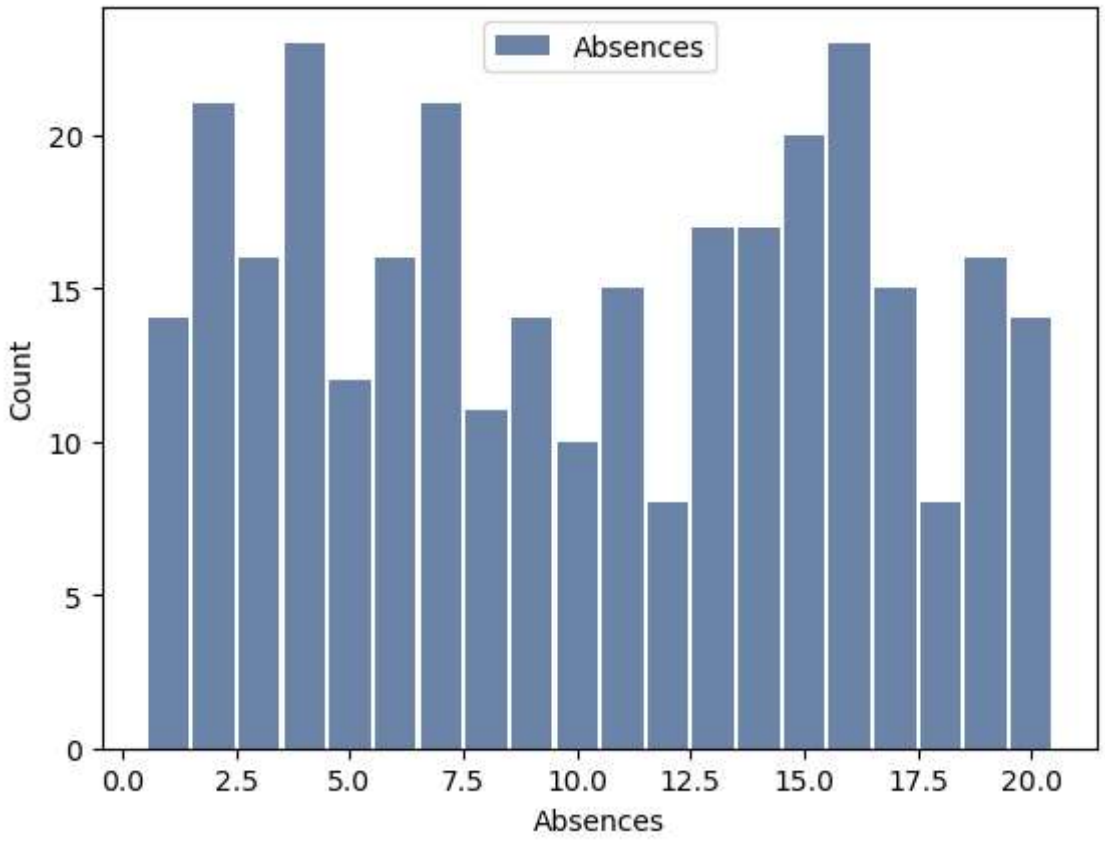
Out[12]:
```
(array([197.,  65.,  32.,   7.,   2.,   3.,   3.,   0.,   1.,   1.]),
 array([ 45046. ,  65541.4,  86036.8, 106532.2, 127027.6, 147523. ,
        168018.4, 188513.8, 209009.2, 229504.6, 250000. ]),
 <BarContainer object of 10 artists>)
```



In [13]:
```
mean_ab = df.Absences.mean()
max_ab = df.Absences.max()
min_ab = df.Absences.min()

mean_ab, max_ab, min_ab
```

Out[13]:
```
(10.237942122186496, 20, 1)
```

In [14]:
```
#This histogram shows the distribution of team member absences. This is not an even di
#throughout the data set. There does not seem to be a mode that really stands out. The
#that after 20 absences, the employee may be subject to termination as there are no fu
#peaks in the dataset may indicate different levels of discapline i.e. written warning
#so on. The mean sits right in the middle of the range at 10.2 absences per team membe
#interresting to see how absences compare to salary, satisfaction ratings, engagement,

hist = thinkstats2.Hist(df.Absences, label='Absences')
thinkplot.Hist(hist)
thinkplot.Config(xlabel='Absences', ylabel='Count')
```

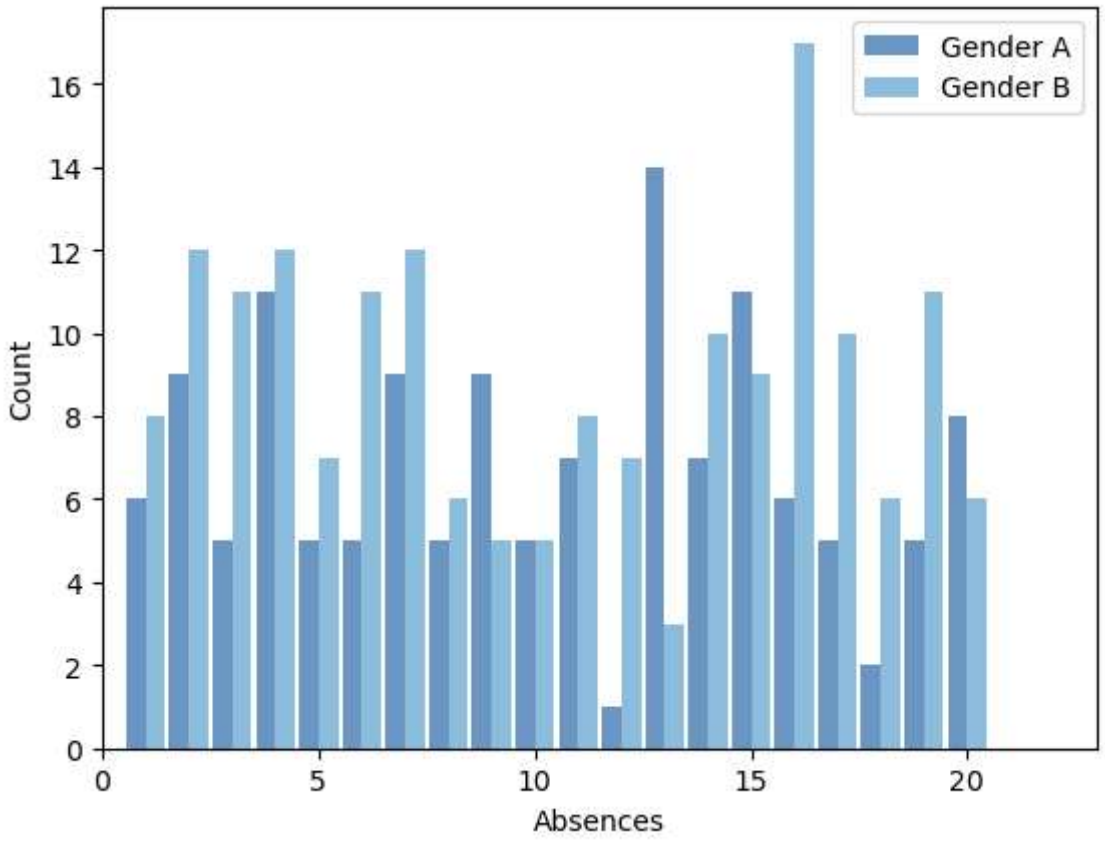


In [15]:
```
#Out of curiousity, i wanted to take a look a the absences variable a bit deeper by ev
#between genders to see if there was a large difference between the two. Results below

gender_a = df[df.GenderID == 1]
gender_b = df[df.GenderID != 1]

gender_a_hist = thinkstats2.Hist(gender_a.Absences, label='Gender A')
gender_b_hist = thinkstats2.Hist(gender_b.Absences , label='Gender B')

total_gen_a_absences = gender_a.Absences.sum()
total_gen_b_absences = gender_b.Absences.sum()

mean_gen_a_absences = gender_a.Absences.mean()
mean_gen_b_absences = gender_b.Absences.mean()

print("Total Gender A Absences = ", total_gen_a_absences)
print("Total Gender B Absences = ", total_gen_b_absences)
print("Mean Gender A Absences = ", mean_gen_a_absences)
print("Mean Gender B Absences = ", mean_gen_b_absences)
```

```
Total Gender A Absences =  1378
Total Gender B Absences =  1806
Mean Gender A Absences =  10.207407407407407
Mean Gender B Absences =  10.261363636363637
```

In [16]:
```
#With this histogram, i split the difference in employee absences between gender A and
#we can see that Gender B has a lot more absences than Gender A, however the mean of b
#distribution below shows that the amount of absences are distributed pretty evenly ur
#10 or more absences. There seems to be a bigger separation between the two at the 12,
#one may speculate what this actually means. Absences can be a result of many reasons,
#trying to compare between genders with this data doesn't really force us to favor one
```

```
width = 0.45
thinkplot.PrePlot(2)
thinkplot.Hist(gender_a_hist, align='right', width=width)
thinkplot.Hist(gender_b_hist, align='left', width=width)
thinkplot.Config(xlabel='Absences', ylabel='Count', xlim=[0, 23])
```



Final Project Bullet 5

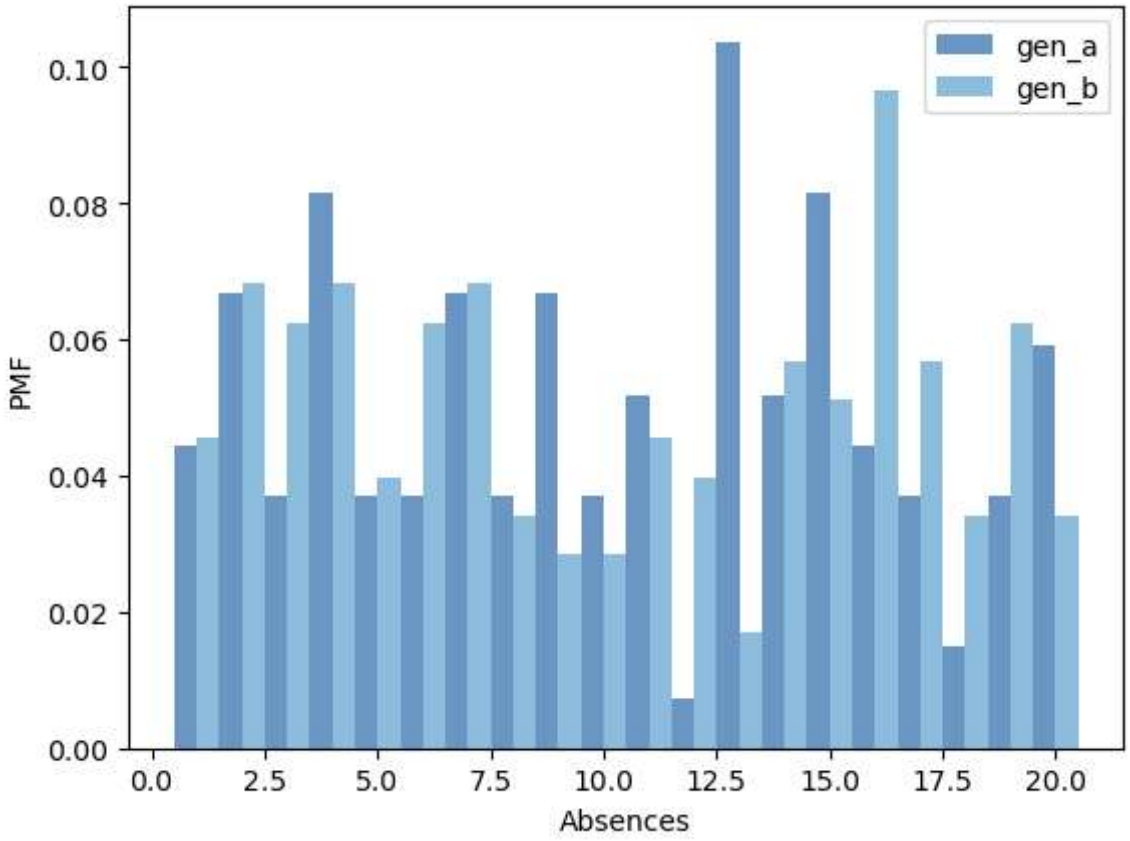
Scenario 1 (Gender / Absences)

Using pg. 29 of your text as an example, compare two scenarios in your data using a PMF. Reminder, this isn't comparing two variables against each other – it is the same variable, but a different scenario. Almost like a filter. The example in the book is first babies compared to all other babies, it is still the same variable, but breaking the data out based on criteria we are exploring (Chapter 3).

In [17]:
```
#below will separate the gender by gender A and gender B and provide the number of dat
#for each.

gen_a_ab = gender_a.Absences
print('Count Gender A', len(gen_a_ab))
print('Mean Absences Gender A', gen_a_ab.mean())

gen_b_ab = gender_b.Absences
print('Count Gender B', len(gen_b_ab))
print('Mean Absences Gender B', gen_b_ab.mean())
```

```
Count Gender A 135
Mean Absences Gender A 10.207407407407407
Count Gender B 176
Mean Absences Gender B 10.261363636363637
```

In [18]:
```
#Using the varable Absences, we will compare two scenarios of gender using PMF.

gender_a_pmf = thinkstats2.Pmf(gen_a_ab, label='gen_a')
gender_b_pmf = thinkstats2.Pmf(gen_b_ab, label='gen_b')

width = 5 / 10

# plot PMFs for gender A and gender B
thinkplot.PrePlot(2)
thinkplot.Hist(gender_a_pmf, align='right', width=width)
thinkplot.Hist(gender_b_pmf, align='left', width=width)
thinkplot.Config(xlabel='Absences', ylabel='PMF')
```



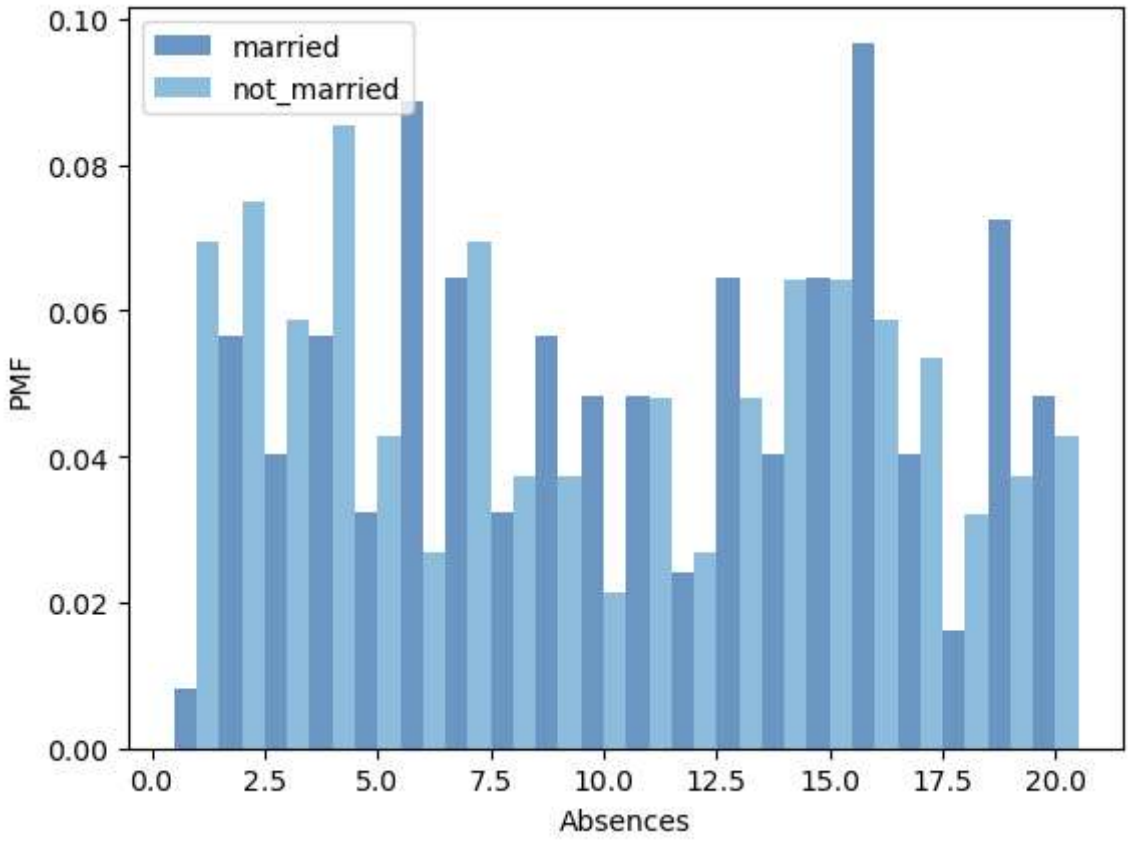
Scenario 2 (Married / Absences)

In [19]:
```
#below will separate the married ID and provide the number of data points for each mar
#for each.

married = df[df.MarriedID == 1]
not_married = df[df.MarriedID != 1]

married_ab = married.Absences
print('Count Married', len(married_ab))
print('Mean Absences Married', married_ab.mean())
```

```
not_married_ab = not_married.Absences
print('Count Not Married', len(not_married_ab))
print('Mean Absences Not Married', not_married_ab.mean())
```

```
Count Married 124
Mean Absences Married 10.92741935483871
Count Not Married 187
Mean Absences Not Married 9.780748663101605
```

In [20]:
```
#Using the varable Absences, we will compare the scenario of married ID using PMF.

married_pmf = thinkstats2.Pmf(married_ab, label='married')
not_married_pmf = thinkstats2.Pmf(not_married_ab, label='not_married')

width = 5 / 10

# plot PMFs for married and not married
thinkplot.PrePlot(2)
thinkplot.Hist(married_pmf, align='right', width=width)
thinkplot.Hist(not_married_pmf, align='left', width=width)
thinkplot.Config(xlabel='Absences', ylabel='PMF', loc='upper left')
```

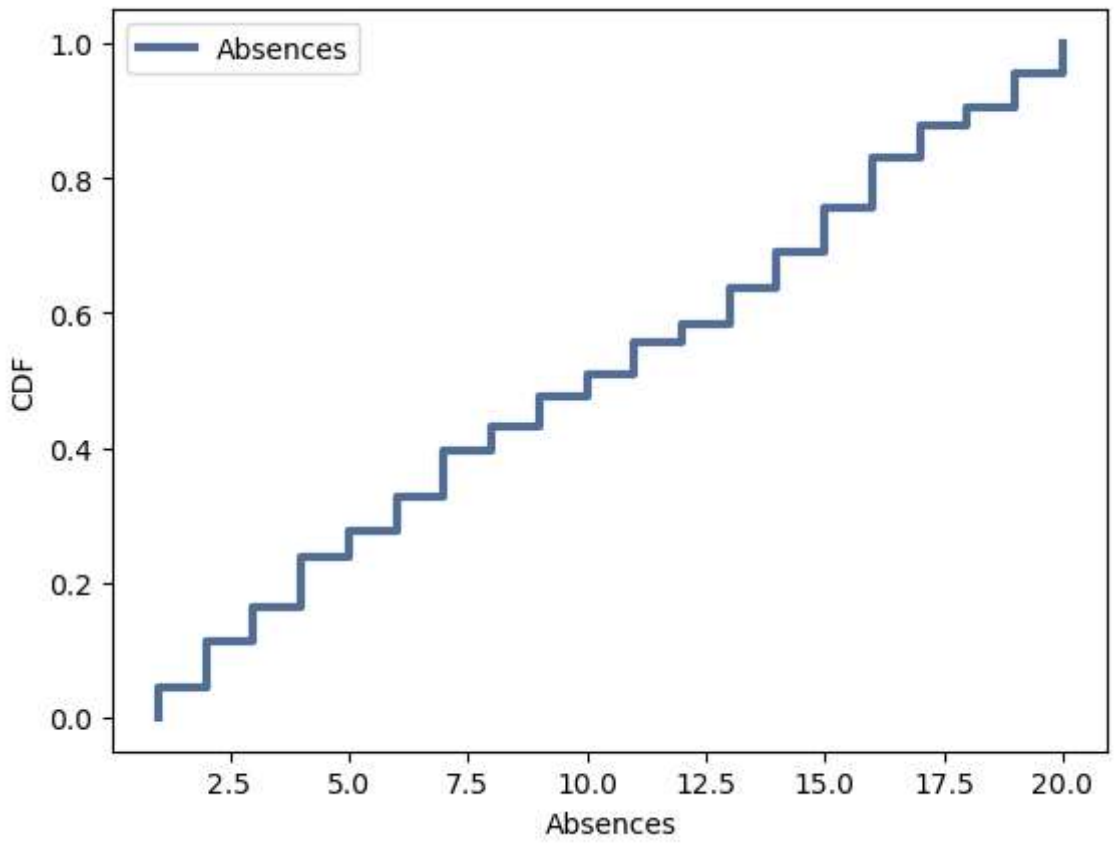


Final Project Bullet 6

Create 1 CDF with one of your variables, using page 41-44 as your guide, what does this tell you about your variable and how does it address the question you are trying to answer (Chapter 4).

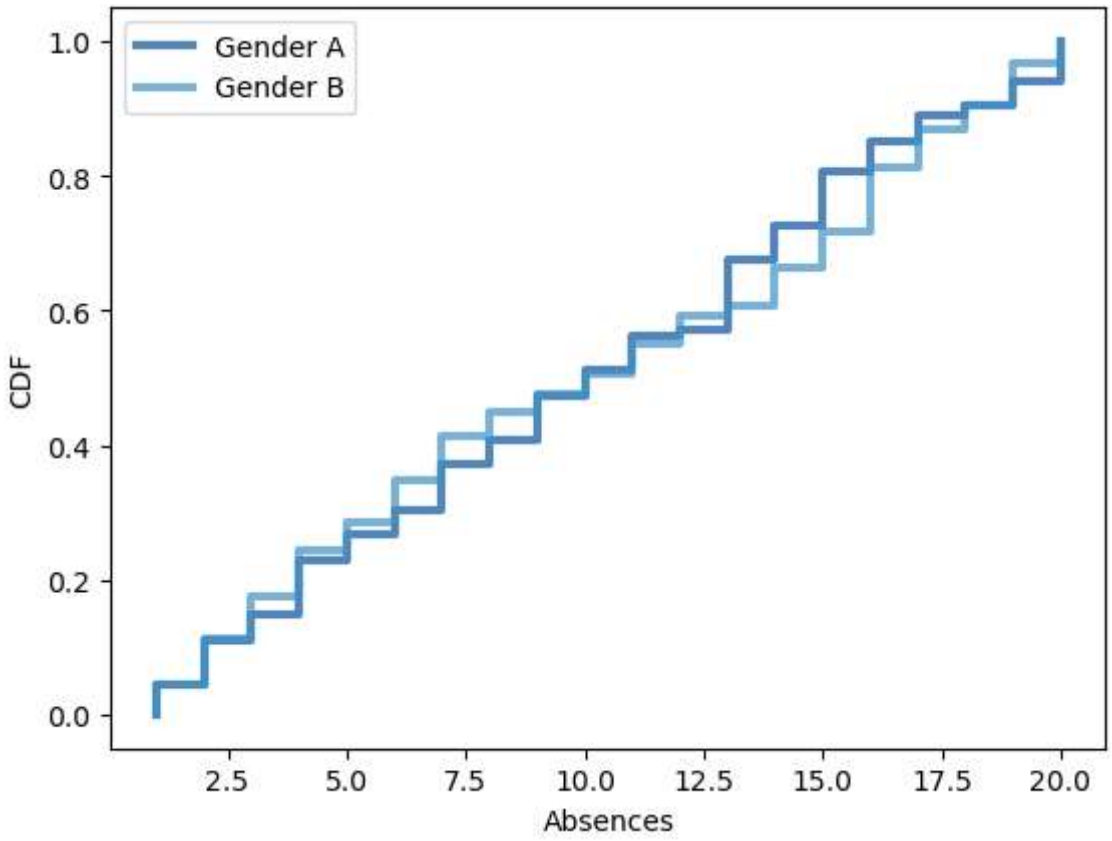
In [21]:
```
#CDF of absences.

cdf = thinkstats2.Cdf(df.Absences, label='Absences')
```

```
thinkplot.Cdf(cdf)
thinkplot.Show(xlabel='Absences', ylabel='CDF', loc='upper left')
```




```
<Figure size 800x600 with 0 Axes>
```

In [22]:
```
#We will perform a comparative CDF of absences between gender a and gender b

gen_a_cdf = thinkstats2.Cdf(gen_a_ab, label='Gender A')
gen_b_cdf = thinkstats2.Cdf(gen_b_ab, label='Gender B')

thinkplot.PrePlot(2)
thinkplot.Cdfs([gen_a_cdf, gen_b_cdf])
thinkplot.Show(xlabel='Absences', ylabel='CDF')
```



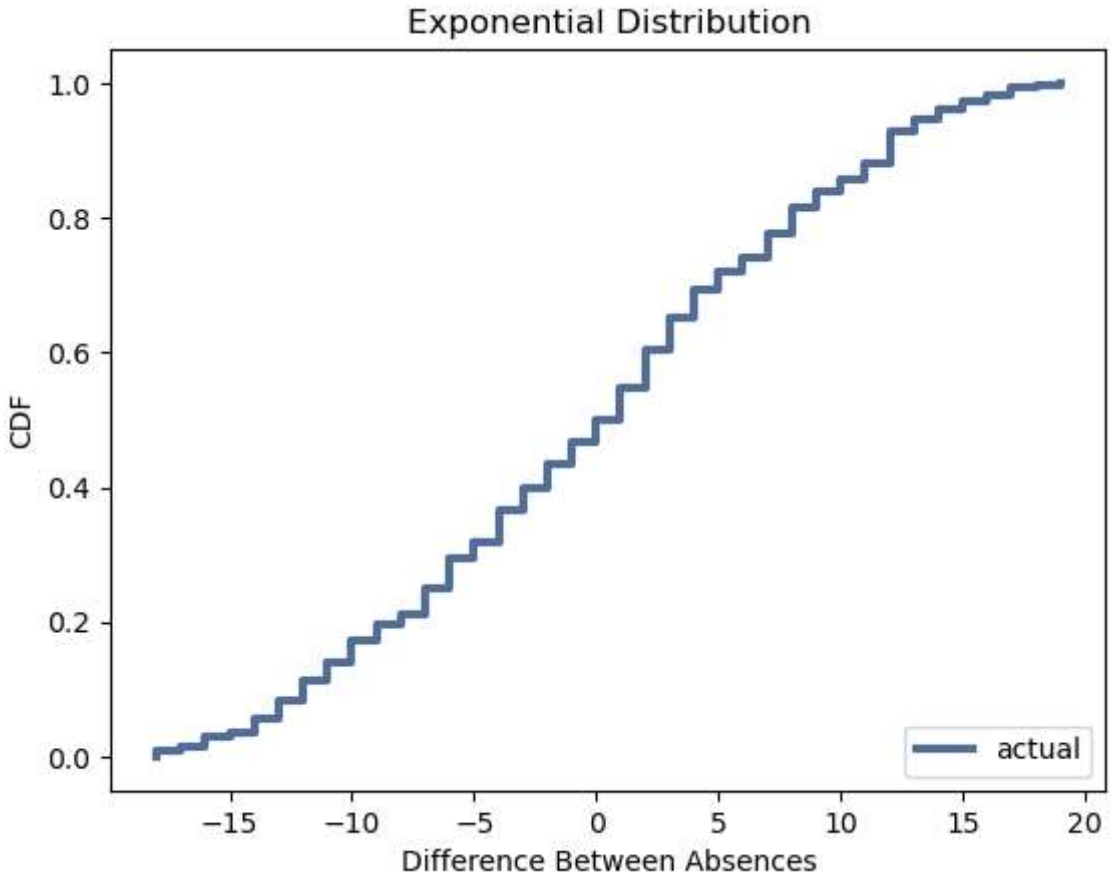

```
<Figure size 800x600 with 0 Axes>
```

This comparison CDF makes the shape of the distribution and the differences between them much clearer. While Gen A seems to have less absences up to about 10 absences, the number of absences rises above gender b above that point. Gender B has a higher absence leve under 10 absences, but seems to increase after that point.

Final Project Bullet 7

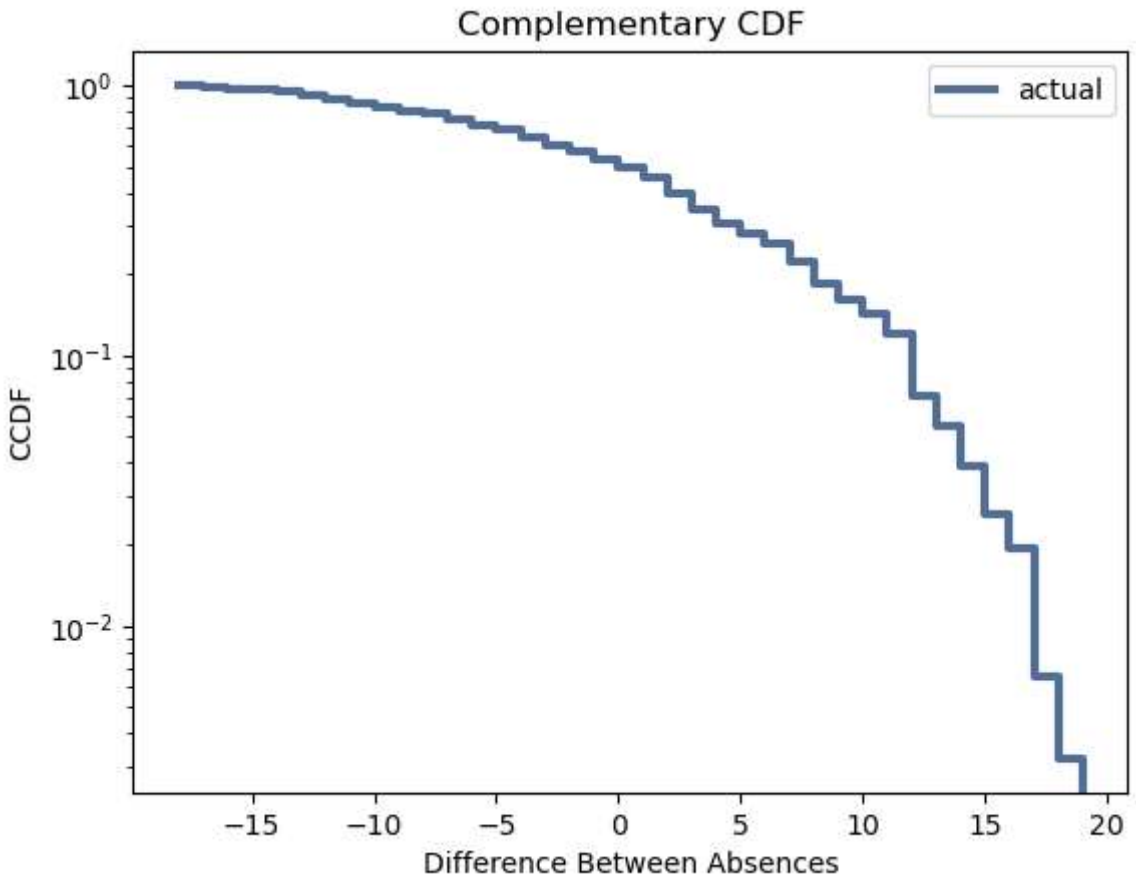
Plot 1 analytical distribution and provide your analysis on how it applies to the dataset you have chosen (Chapter 5).

In [23]:
```
#I chose exponetial distribution because I wanted to see if the series of events are e
#make an curve shape of an exponetial distribution. Plotting below shows a slight expo

import analytic

diffs = df.Absences.diff()
cdf = thinkstats2.Cdf(diffs, label="actual")

thinkplot.Cdf(cdf)
thinkplot.Config(xlabel="Difference Between Absences", ylabel="CDF", loc="lower right'
```



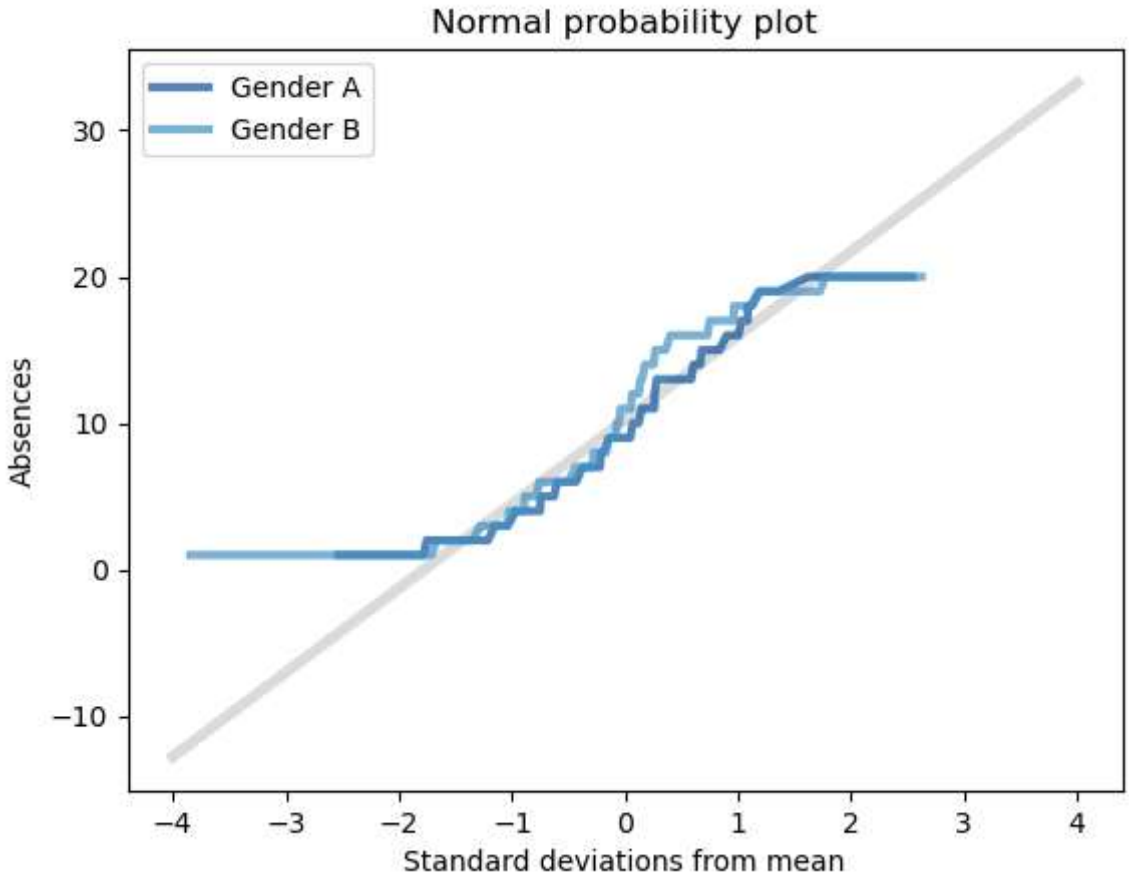
In [24]:

```
#We check the exponential distribution by using a complimentary CDF. For data from an
#the result is a straight line. In the figure below, the results is not exactly straig
#the exponential distribution for absences may not be the perfect model for this data.

thinkplot.Cdf(cdf, complement=True)
thinkplot.Config(
    xlabel="Difference Between Absences",
    ylabel="CCDF",
    yscale="log",
    title="Complementary CDF",
    loc="upper right",
)
```



In [25]:

```
#I decided to also try the normal probability plot to compare Gen A and Gen B absences

absences = df.Absences

mean, var = thinkstats2.TrimmedMeanVar(absences, p=0.01)
std = np.sqrt(var)

xs = [-4, 4]
fxs, fys = thinkstats2.FitLine(xs, mean, std)
thinkplot.Plot(fxs, fys, linewidth=4, color="0.8")

thinkplot.PrePlot(2)
xs, ys = thinkstats2.NormalProbability(gen_a_ab)
thinkplot.Plot(xs, ys, label="Gender A")

xs, ys = thinkstats2.NormalProbability(gen_b_ab)
thinkplot.Plot(xs, ys, label="Gender B")
thinkplot.Config(
    title="Normal probability plot",
    xlabel="Standard deviations from mean",
    ylabel="Absences",
    loc="upper left"
)

#The figure below shows results for both Gender A and Gender B. Both curves match the
#at the tails. Both genders curves are slightly lower than what the model expects belo
#higher than what the model expects above the mean.
#The plot suggest thtat the normal model describes the distribution well for both Geno
#within a few standard deviations fom the mean, but not within the tails.
```



Final Project Bullet 8

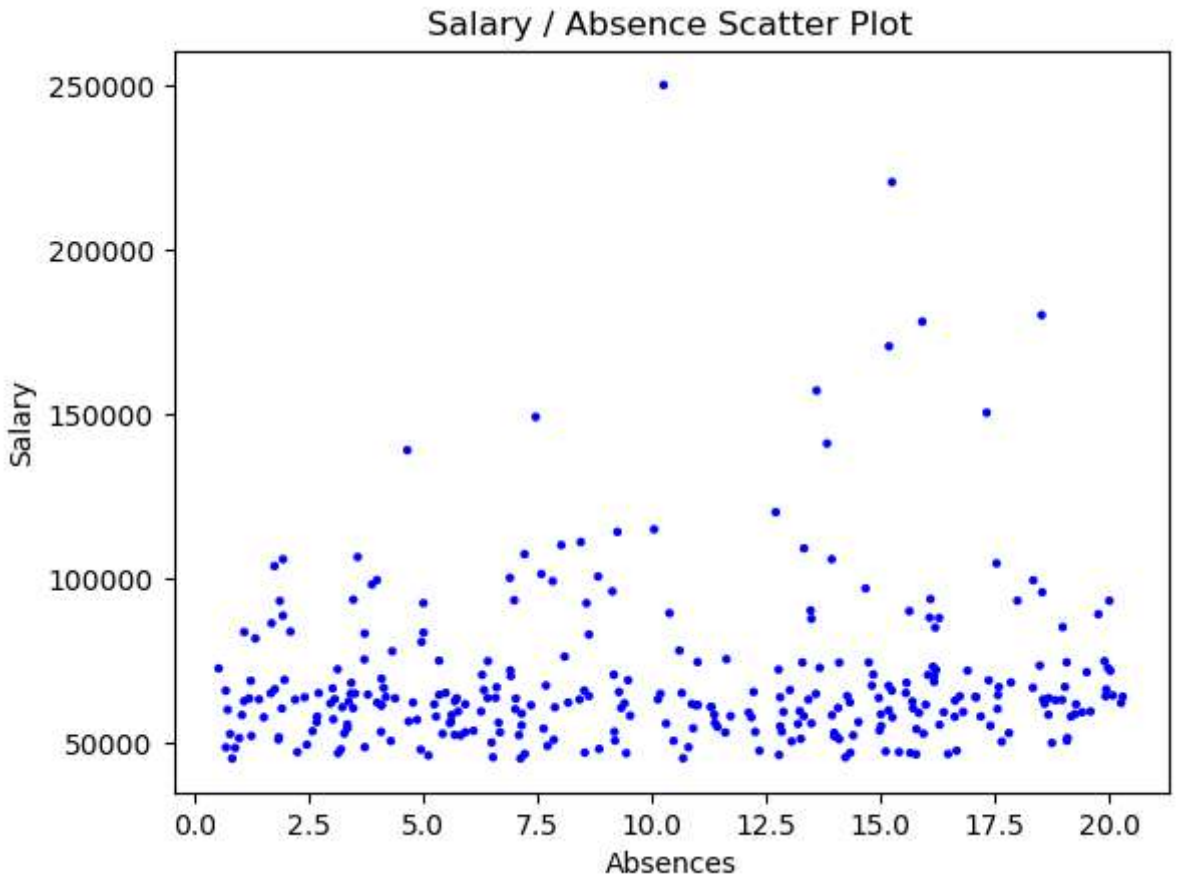
Create two scatter plots comparing two variables and provide your analysis on correlation and causation. Remember, covariance, Pearson's correlation, and Non-Linear Relationships should also be considered during your analysis (Chapter 7).

Scatter Plot 1 (absences and salary)

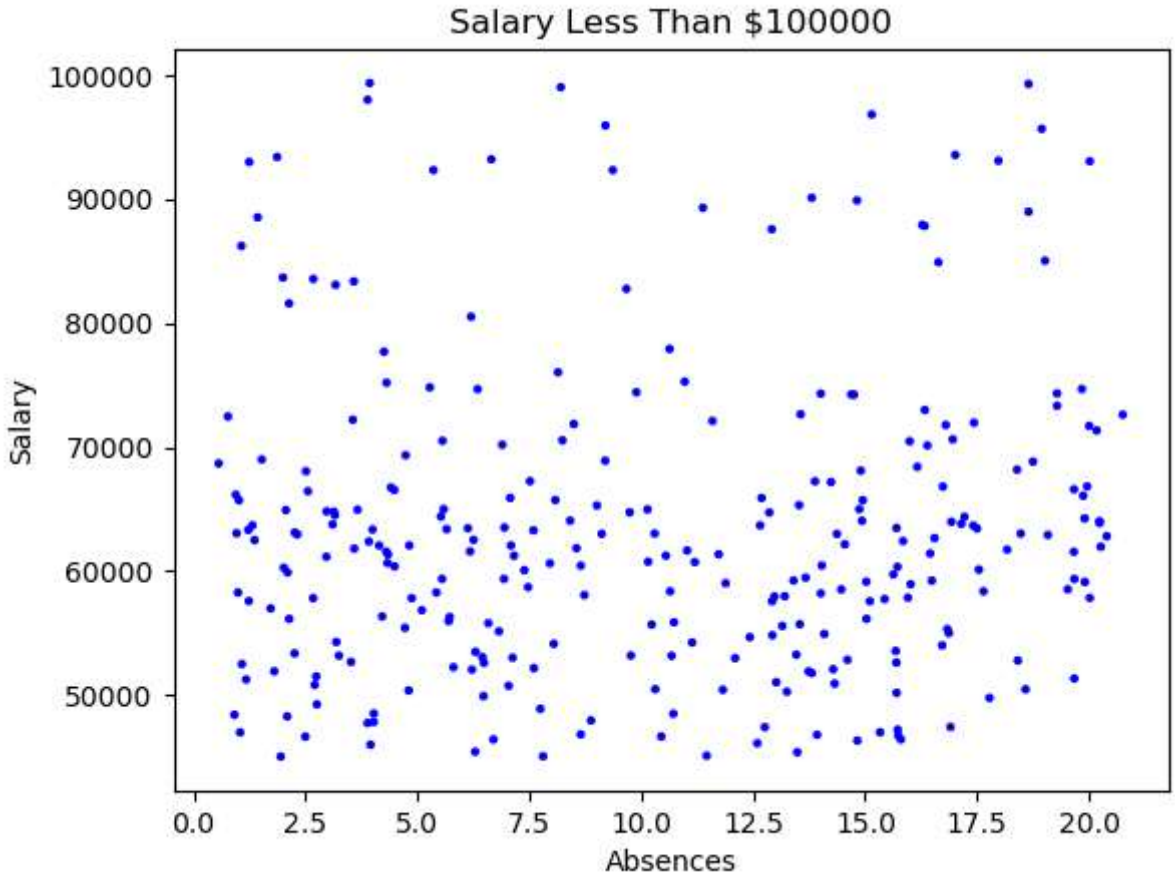
We will review to see if there is a correlation between salary and absences. One may think team members with better salary have better attendance.

In [26]:
```
#The model below shows some grouping around the $50000 to $100000 salary range, howeve
#linear relationship because the majority of the employees are at this salary level. T
#remove some of the outliers or the upper salary level.

salary = df.Salary

def Jitter(values, jitter=0.5):
    n = len(values)
    return np.random.normal(0, jitter, n) + values

salary1 = Jitter(salary, 0.5)
absences1 = Jitter(absences, 0.5)
```

```
thinkplot.Scatter(absences1, salary1, alpha=1.0, s=10)
thinkplot.Config(xlabel='Absences',
                 ylabel='Salary',
                 title="Salary / Absence Scatter Plot",
                 legend=False)
```



In [27]:
```
#In this scatter ploty we filter to the salary income below $100000 which is where the
#Examining the data, do not see much linear relationship in the scatter plot.

lsalary = df[df.Salary < 100000]

def Jitter(values, jitter=0.5):
    n = len(values)
    return np.random.normal(0, jitter, n) + values

salary2 = Jitter(lsalary.Salary, 0.5)
absences2 = Jitter(lsalary.Absences, 0.5)

thinkplot.Scatter(absences2, salary2, alpha=1.0, s=10)
thinkplot.Config(xlabel='Absences',
                 ylabel='Salary',
                 title='Salary Less Than $100000',
                 legend=False)
```



In [28]:
```
def Cov(xs, ys, meanx=None, meany=None):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    if meanx is None:
        meanx = np.mean(xs)
    if meany is None:
        meany = np.mean(ys)

    cov = np.dot(xs-meanx, ys-meany) / len(xs)
    return cov

#Pearson's Correlation

def Corr(xs, ys):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    meanx, varx = thinkstats2.MeanVar(xs)
    meany, vary = thinkstats2.MeanVar(ys)

    corr = Cov(xs, ys, meanx, meany) / np.sqrt(varx * vary)
    return corr
```

In [29]:
```
#Covariance is the measure of the tendancy of two variables to vary together. Below we
#Correlation is a statistic intended to quanify the strength of the reltionship betwee
#The correlation below is 0.066 which is a relatively weak correlation between absence
#Pearson's Correlation, there is no linear relationship between these two variables.

Cov1 = Cov(absences2, salary2)
```

```
PCorr1 = Corr(absences2, salary2)
print('Covariance = ', Cov1)
print('Pearson Correlation = ', PCorr1)
```

```
Covariance =  4583.829743146342
Pearson Correlation =  0.06258296507656769
```

In [30]:
```
#Spearman Correlation Function

def SpearmanCorr(xs, ys):
    xranks = pd.Series(xs).rank()
    yranks = pd.Series(ys).rank()
    return Corr(xranks, yranks)
```

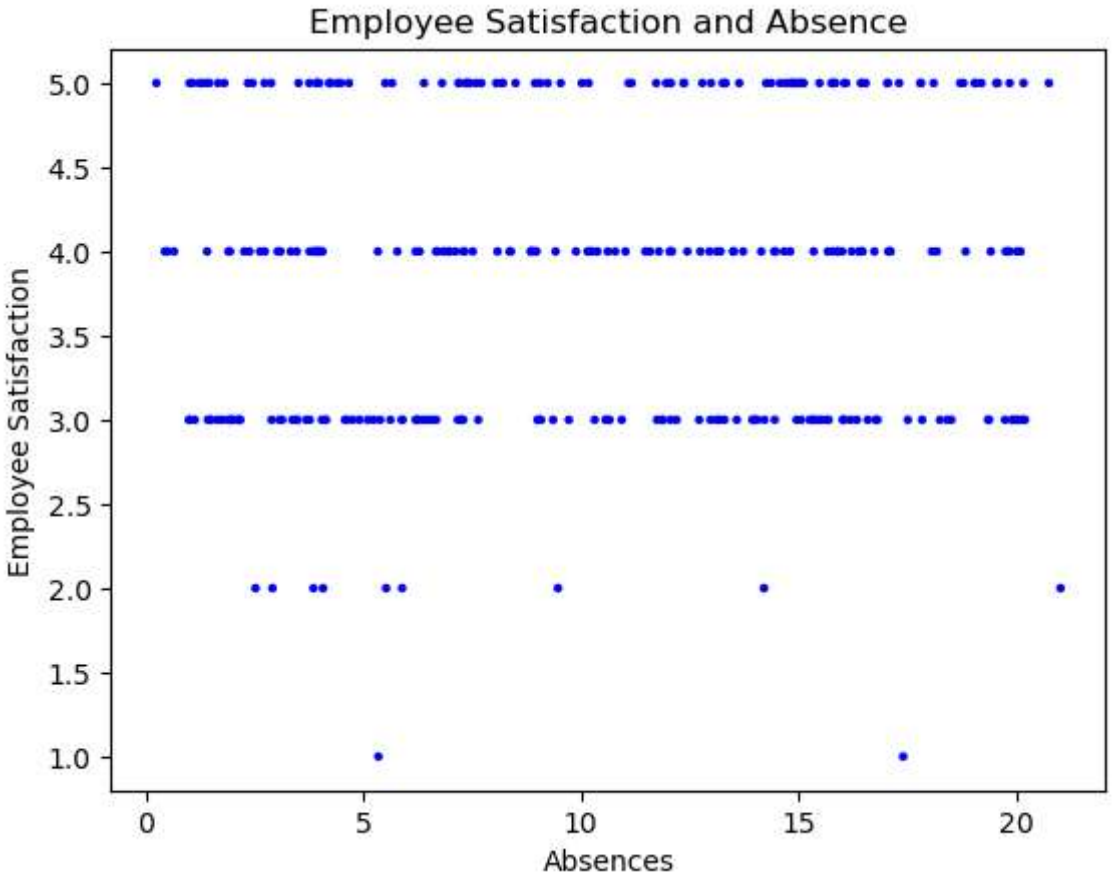
In [31]:
```
#Spearman's Correlation of the lower income salary is slightly higher than the Pearsor
#Both indicate a weak correlation between Absences and Salary or at least no linear re
#Therefore, one may conclude that there is no or a week relatationship between income

SCorr = SpearmanCorr(absences2, salary2)
print('Spearman Correlation = ', SCorr)
```

```
Spearman Correlation =  0.0787081980920235
```

Scatter Plot 2 (Absences and Employee Satisfaction Rating)

Perhaps there is a correlation between absences and how satisfied a team member is with their job. One would think that employees more satisifed would have better attendance.

In [32]:
```
#Review of the scatter plot below does not show much linear relationship between emplo
#Team members with high satisfactory ratings seemed to have a similar distribution of
#medium or medium high employee satisfactory ratings.

empsatis = df.EmpSatisfaction

def Jitter(values, jitter=0.5):
    n = len(values)
    return np.random.normal(0, jitter, n) + values

empsatis1 = Jitter(empsatis, 0.0)
absences1 = Jitter(absences, 0.5)

thinkplot.Scatter(absences1, empsatis1, alpha=1.0, s=10)
thinkplot.Config(xlabel='Absences',
                 ylabel='Employee Satisfaction',
                 title='Employee Satisfaction and Absence',
                 legend=False)
```



In [33]:
```
#Below we see a positive covariance of 0.42 indicating a positive reaction between the
#Correlation is a statistic intended to quanify the strength of the reltionship betwee
#The correlation below is 0.078 which is a relatively weak correlation between absence
#Pearson's Correlation, there is no linear relationship between these two variables.

Cov2 = Cov(absences1, empsatis1)
PCorr2 = Corr(absences1, empsatis1)
print('Covariance = ', Cov2)
print('Pearson Correlation = ', PCorr2)
```

```
Covariance =  0.3621110253547405
Pearson Correlation =  0.06807657447254341
```

In [34]:
```
#Spearman's Correlation of the lower income salary is slightly higher than the Pearsor
#Both indicate a weak correlation between Absences and Salary or at least no linear re
#Therefore, one may conclude that there is no or a week relatationship between employe
#Apparently even those employees that are completely satisfied with their job still mi
#those who are not satisfied.

SCorr = SpearmanCorr(absences1, empsatis1)
print('Spearman Correlation = ', SCorr)
```

```
Spearman Correlation =  0.0692230503320501
```

Final Project Bullet 9

Conduct a test on your hypothesis using one of the methods covered in Chapter 9. The method I will use is "Testing a Correlation" on page 107

In [35]:
```
class HypothesisTest(object):

    def __init__(self, data):
        self.data = data
        self.MakeModel()
        self.actual = self.TestStatistic(data)

    def PValue(self, iters=1000):
        self.test_stats = [self.TestStatistic(self.RunModel())
                           for _ in range(iters)]

        count = sum(1 for x in self.test_stats if x >= self.actual)
        return count / iters

    def TestStatistic(self, data):
        raise UnimplementedMethodException()

    def MakeModel(self):
        pass

    def RunModel(self):
        raise UnimplementedMethodException()
```

In [36]:
```
class CorrelationPermute(thinkstats2.HypothesisTest):

    def TestStatistic(self, data):
        xs, ys = data
        test_stat = abs(thinkstats2.Corr(xs, ys))
        return test_stat

    def RunModel(self):
        xs, ys = self.data
        xs = np.random.permutation(xs)
        return xs, ys
```

In [37]:
```
#The pValue between Engagement Survey and Absences Correlation test shows a relatively
#weak correlation between the engagement survey and absences and therefore not statist
#However, this may be in part that the engagement survey has a max value of 5.0 and te
#all of the survey will reach a max value and may still have a large distribution in c


cleaned = df.dropna(subset=['EngagementSurvey', 'Absences'])
data = cleaned.EngagementSurvey.values, cleaned.Absences.values
ht = CorrelationPermute(data)
pvalue = ht.PValue()
pvalue
```

Out[37]:
```
0.852
```

Final Project Bullet 10

For this project, conduct a regression analysis on either one dependent and one explanatory variable, or multiple explanatory variables (Chapter 10 & 11).

In [38]:
```
import statsmodels.formula.api as smf

formula = 'Absences ~ EngagementSurvey + GenderID + Salary + DeptID + EmpSatisfaction
results = smf.ols(formula, data=df).fit()
results.summary()
```

Out[38]:

OLS Regression Results

Dep. Variable:	Absences	**R-squared:**	0.044
Model:	OLS	**Adj. R-squared:**	0.018
Method:	Least Squares	**F-statistic:**	1.709
Date:	Fri, 11 Aug 2023	**Prob (F-statistic):**	0.0958
Time:	03:08:24	**Log-Likelihood:**	-959.53
No. Observations:	303	**AIC:**	1937.
Df Residuals:	294	**BIC:**	1970.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	3.9746	3.561	1.116	0.265	-3.034	10.983
EngagementSurvey	-0.1892	0.436	-0.434	0.665	-1.048	0.669
GenderID	-0.0454	0.680	-0.067	0.947	-1.384	1.293
Salary	4.125e-05	1.59e-05	2.597	0.010	9.99e-06	7.25e-05
DeptID	0.2381	0.520	0.458	0.647	-0.786	1.262
EmpSatisfaction	0.2411	0.384	0.629	0.530	-0.514	0.996
MaritalStatusID	0.2022	0.358	0.565	0.573	-0.503	0.907
ManagerID	0.1353	0.052	2.582	0.010	0.032	0.238
SpecialProjectsCount	0.0497	0.243	0.204	0.838	-0.429	0.528

Omnibus:	152.906	**Durbin-Watson:**	2.032
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	18.004
Skew:	0.058	**Prob(JB):**	0.000123
Kurtosis:	1.812	**Cond. No.**	7.91e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.91e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Running multiple regression, only salary seems to have an extremely low pValue indicating statistical significance, however we know that there is no linear relationship with the model and the distribution of salary is quite large with several outliers on the high end.

We can take a closer look at salary by filtering out the outliers with greater than 100000 income

In [39]:
```
#Filtering out the outliers in salary and focusing on salary and absences from team me

salary3 = lsalary.Salary
absences3 = lsalary.Absences

formula = 'absences3 ~ salary3'
model = smf.ols(formula, data=lsalary)
results = model.fit()
results.summary()
```

Out[39]:

OLS Regression Results

Dep. Variable:	absences3	**R-squared:**	0.004
Model:	OLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	1.257
Date:	Fri, 11 Aug 2023	**Prob (F-statistic):**	0.263
Time:	03:08:24	**Log-Likelihood:**	-914.51
No. Observations:	286	**AIC:**	1833.
Df Residuals:	284	**BIC:**	1840.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	8.1949	1.843	4.446	0.000	4.567	11.823
salary3	3.203e-05	2.86e-05	1.121	0.263	-2.42e-05	8.83e-05

Omnibus:	476.096	**Durbin-Watson:**	2.089
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	21.459
Skew:	0.011	**Prob(JB):**	2.19e-05
Kurtosis:	1.658	**Cond. No.**	3.38e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.38e+05. This might indicate that there are strong multicollinearity or other numerical problems.

The pValue comes much closer and is now positive, however still does not show any statistical significance.

In [ ]: