Data Science 1000 Assignment 2

Teagan Martins

1.

a) 3.03 = 3.25 – 0.22 (standard deviation) therefore 3.03 is 1 standard deviation below the mean.

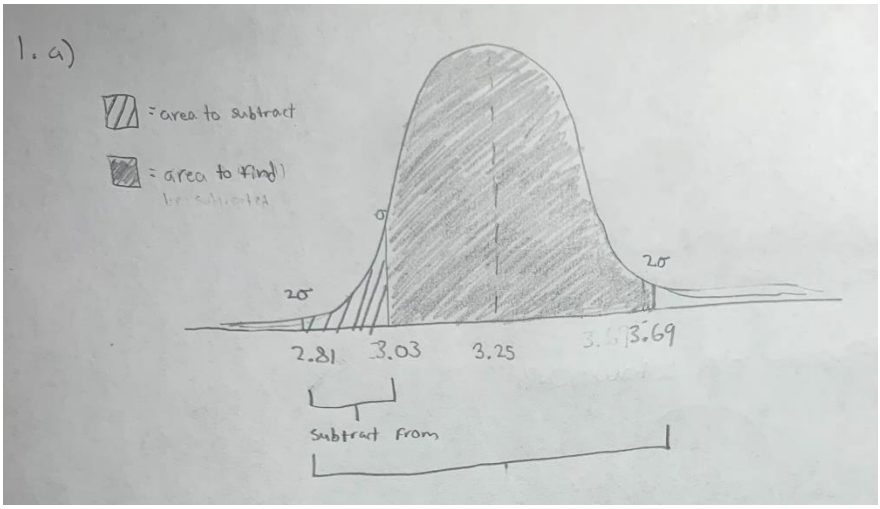
3.69 = 3.25 + 2 x 0.22 (standard deviation) therefore 3.69 is 2 standard deviations above the mean.

2.81 = 3.25 – 2 x 0.22 therefore 2.81 is 2 standard deviations below the mean.

The area between 2.81 and 3.69 makes up 95% of distributions. I must subtract the small area between 2.81 and 3.03.

The area between 3.03 and 3.25 makes up 34% of distributions whereas the area between 2.81 and 3.25 makes up 47.5% of distributions. 47.5 – 34 = 13.5%. The small area makes up 13.5% of distributions.

95 – 13.5 = 81.5%. 81.5% of candidates have a GPA between 3.03 and 3.69.

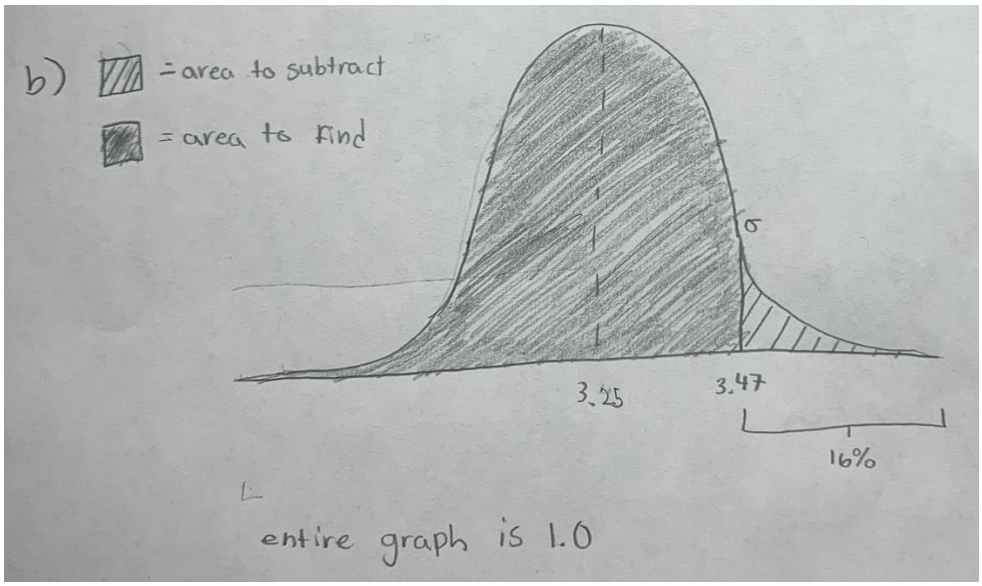


b) 3.47 = 3.25 + 0.22 therefore 3.47 is 1 standard deviation above the mean.

3.47 marks the 68% mark. The area above 3.47 is 16% (32/2). The total area of the entire graph is 1.0.

1.0 – 0.16 = 0.84 = 84%.

84% of the candidates have a GPA below 3.47.



c) Top 5% = 95% below them.

A z-score between 1.65 and 1.64 represents 95% of scores below it. We can assume the z-score would be 1.645.

$1.645 = (x - 3.25) / 0.22$

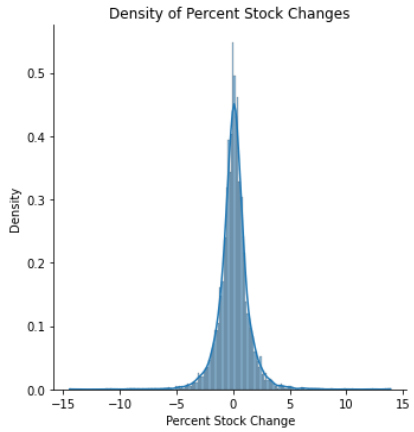
$0.3619 = x - 3.25$

$3.6119 = x$

To be in the top 5% of all successful interviewees, a candidate would have to have a GPA of 3.6119.

2. a) The shape of the distribution is a symmetrical bell curve. It looks very similar to a normal curve.

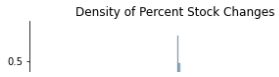
In [61]:
```
# Question 2a,b,c

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile

rbc_csv = pandas.read_csv('RBC_Returns.csv')

stock_change = rbc_csv['percent_stock_change']
dates = rbc_csv['date']
sns.displot(stock_change,kde=True, stat="density")
plt.ylabel("Density")
plt.xlabel("Percent Stock Change")
plt.title("Density of Percent Stock Changes")
plt.show()

u = stock_change.mean()
print("The mean is " + str(u))
s = numpy.std(stock_change)
print("The standard deviation is " + str(s))

count = 0

for x in stock_change:
    if (x > (u - s) and x < (u + s)):
        count = count + 1
print(str(count) + " data points have a % daily change between mean - s and mean + s")

values = len(stock_change)

proportion = (count/values)
print(str(proportion) + " or " + str(proportion*100) + "% of data points have a % daily change between mean - s and mean + s")
```

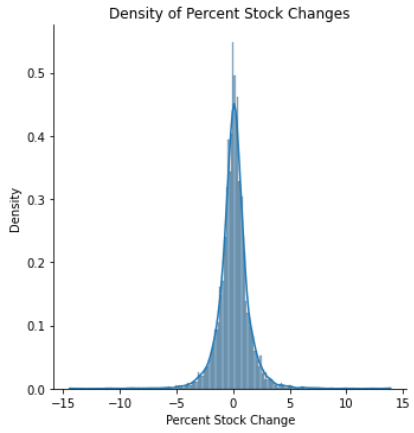



```
The mean is 0.058340604526355304
The standard deviation is 1.3957913093573147
5357 data points have a % daily change between mean - s and mean + s
0.7976474091721263 or 79.76474091721263% of data points have a % daily change between mean - s and mean + s
```

b)

In [61]:
```
# Question 2a,b,c

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile

rbc_csv = pandas.read_csv('RBC_Returns.csv')

stock_change = rbc_csv['percent_stock_change']
dates = rbc_csv['date']
sns.displot(stock_change,kde=True, stat="density")
plt.ylabel("Density")
plt.xlabel("Percent Stock Change")
plt.title("Density of Percent Stock Changes")
plt.show()

u = stock_change.mean()
print("The mean is " + str(u))
s = numpy.std(stock_change)
print("The standard deviation is " + str(s))

count = 0

for x in stock_change:
    if (x > (u - s) and x < (u + s)):
        count = count + 1
print(str(count) + " data points have a % daily change between mean - s and mean + s")

values = len(stock_change)

proportion = (count/values)
print(str(proportion) + " or " + str(proportion*100) + "% of data points have a % daily change between mean - s and mean + s")
```

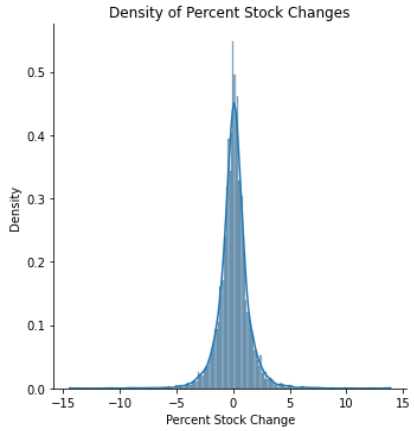





```
The mean is 0.058340604526355304
The standard deviation is 1.3957913093573147
5357 data points have a % daily change between mean - s and mean + s
0.7976474091721263 or 79.76474091721263% of data points have a % daily change between mean - s and mean + s
```

c) Yes, there does seem to be a departure from the normal distribution as the proportion of values within 1 standard deviation represents 79.67% of all the values. In a normal distribution, this would be 68%.

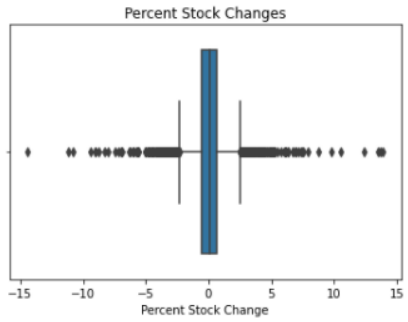
In [61]:
```
# Question 2a,b,c

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile

rbc_csv = pandas.read_csv('RBC_Returns.csv')

stock_change = rbc_csv['percent_stock_change']
dates = rbc_csv['date']
sns.displot(stock_change,kde=True, stat="density")
plt.ylabel("Density")
plt.xlabel("Percent Stock Change")
plt.title("Density of Percent Stock Changes")
plt.show()

u = stock_change.mean()
print("The mean is " + str(u))
s = numpy.std(stock_change)
print("The standard deviation is " + str(s))

count = 0

for x in stock_change:
    if (x > (u - s) and x < (u + s)):
        count = count + 1
print(str(count) + " data points have a % daily change between mean - s and mean + s")

values = len(stock_change)

proportion = (count/values)
print(str(proportion) + " or " + str(proportion*100) + "% of data points have a % daily change between mean - s and mean + s")
```




```
The mean is 0.058340604526355304
The standard deviation is 1.3957913093573147
5357 data points have a % daily change between mean - s and mean + s
0.7976474091721263 or 79.76474091721263% of data points have a % daily change between mean - s and mean + s
```

d) Yes, my boxplot does support my findings because the graph is not completely symmetrical as seen by the numerous outliers. If the boxplot was completely symmetrical, it would be a normal distribution.

In [65]:
```
# Question 2d

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile

rbc_csv = pandas.read_csv('RBC_Returns.csv')

stock_change = rbc_csv['percent_stock_change']
dates = rbc_csv['date']

sns.boxplot(x = stock_change)
plt.xlabel("Percent Stock Change")
plt.title("Percent Stock Changes")

plt.show()
```



3. a)

In [75]:
```
# Question 3a,c,e,f

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile
from scipy.stats import norm

print(norm.cdf(0.6, loc = 0.8, scale = 0.078), end='')
print(' or ',end='')
print(norm.cdf(0.6, loc = 0.8, scale = 0.078)*100, end='')
print('% of flies have a thorax length less than 0.6mm')

print()

print(1 - norm.cdf(0.9, loc = 0.8, scale = 0.078), end='')
print(' or ', end='')
print((1 - norm.cdf(0.9, loc = 0.8, scale = 0.078))*100, end='')
print('% of flies have a thorax length greater than 0.9mm')

x = (norm.cdf(0.9, loc = 0.8, scale = 0.078))
y = (norm.cdf(0.6, loc = 0.8, scale = 0.078))

print()

print(str(x-y) + ' or ' + str((x-y)*100) + '% of flies have a thorax length between 0.6mm and 0.9mm')

print()

print(str(norm.ppf(0.25,0.8, 0.078)) + 'mm thorax length gives a proportion of 25% of flies below it')
```

```
0.005172148601499026 or 0.5172148601499027% of flies have a thorax length less than 0.6mm

0.09991232866619182 or 9.991232866619182% of flies have a thorax length greater than 0.9mm

0.8949155227323091 or 89.49155227323091% of flies have a thorax length between 0.6mm and 0.9mm

0.747389794847056mm thorax length gives a proportion of 25% of flies below it
```

b) To calculate the proportion in part a) if I only had access to Table A in my notebook, I would use the z-score formula. I would input the value, the mean, and the standard deviation which would end up as (0.6-0.8)/0.078. This results in a z-score of -2.564102564 which we can round to-2.56. Then, I'd look up the z-score value and find the matching percent. In Table A, this results in 0.0052 or 0.52%.

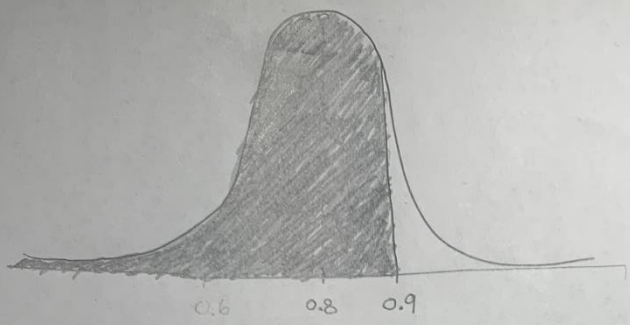
c)

In [75]:
```
# Question 3a,c,e,f

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile
from scipy.stats import norm

print(norm.cdf(0.6, loc = 0.8, scale = 0.078), end='')
print(' or ',end='')
print(norm.cdf(0.6, loc = 0.8, scale = 0.078)*100, end='')
print('% of flies have a thorax length less than 0.6mm')

print()

print(1 - norm.cdf(0.9, loc = 0.8, scale = 0.078), end='')
print(' or ', end='')
print((1 - norm.cdf(0.9, loc = 0.8, scale = 0.078))*100, end='')
print('% of flies have a thorax length greater than 0.9mm')

x = (norm.cdf(0.9, loc = 0.8, scale = 0.078))
y = (norm.cdf(0.6, loc = 0.8, scale = 0.078))

print()

print(str(x-y) + ' or ' + str((x-y)*100) + '% of flies have a thorax length between 0.6mm and 0.9mm')

print()

print(str(norm.ppf(0.25,0.8, 0.078)) + 'mm thorax length gives a proportion of 25% of flies below it')
```

```
0.005172148601499027 or 0.5172148601499027% of flies have a thorax length less than 0.6mm

0.09991232866619182 or 9.991232866619182% of flies have a thorax length greater than 0.9mm

0.8949155227323091 or 89.49155227323091% of flies have a thorax length between 0.6mm and 0.9mm

0.7473897994847056mm thorax length gives a proportion of 25% of flies below it
```

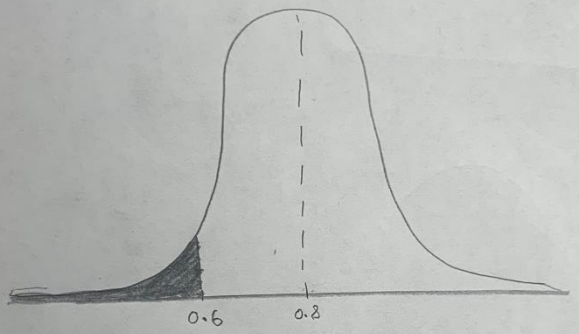
d)

3d)



Calculate proportion below 0.9

(subtract)



Calculate proportion below 0.6

Subtract proportion below 0.6 from the proportion below 0.9.

e)

In [75]:
```
# Question 3a,c,e,f

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile
from scipy.stats import norm

print(norm.cdf(0.6, loc = 0.8, scale = 0.078), end='')
print(' or ',end='')
print(norm.cdf(0.6, loc = 0.8, scale = 0.078)*100, end='')
print('% of flies have a thorax length less than 0.6mm')

print()

print(1 - norm.cdf(0.9, loc = 0.8, scale = 0.078), end='')
print(' or ', end='')
print((1 - norm.cdf(0.9, loc = 0.8, scale = 0.078))*100, end='')
print('% of flies have a thorax length greater than 0.9mm')

x = (norm.cdf(0.9, loc = 0.8, scale = 0.078))
y = (norm.cdf(0.6, loc = 0.8, scale = 0.078))

print()

print(str(x-y) + ' or ' + str((x-y)*100) + '% of flies have a thorax length between 0.6mm and 0.9mm')

print()

print(str(norm.ppf(0.25,0.8, 0.078)) + 'mm thorax length gives a proportion of 25% of flies below it')
```

```
0.005172148601499027 or 0.5172148601499027% of flies have a thorax length less than 0.6mm

0.09991232866619182 or 9.991232866619182% of flies have a thorax length greater than 0.9mm

0.8949155227323091 or 89.49155227323091% of flies have a thorax length between 0.6mm and 0.9mm

0.7473897994847056mm thorax length gives a proportion of 25% of flies below it
```

f)

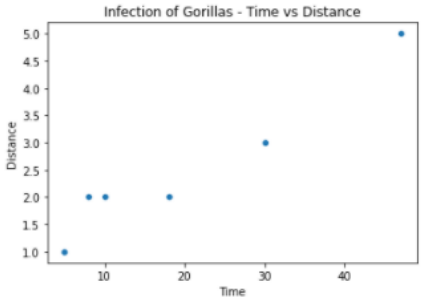
In [75]:
```
# Question 3a,c,e,f

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile
from scipy.stats import norm

print(norm.cdf(0.6, loc = 0.8, scale = 0.078), end='')
print(' or ',end='')
print(norm.cdf(0.6, loc = 0.8, scale = 0.078)*100, end='')
print('% of flies have a thorax length less than 0.6mm')

print()

print(1 - norm.cdf(0.9, loc = 0.8, scale = 0.078), end='')
print(' or ', end='')
print((1 - norm.cdf(0.9, loc = 0.8, scale = 0.078))*100, end='')
print('% of flies have a thorax length greater than 0.9mm')

x = (norm.cdf(0.9, loc = 0.8, scale = 0.078))
y = (norm.cdf(0.6, loc = 0.8, scale = 0.078))

print()

print(str(x-y) + ' or ' + str((x-y)*100) + '% of flies have a thorax length between 0.6mm and 0.9mm')

print()

print(str(norm.ppf(0.25,0.8, 0.078)) + 'mm thorax length gives a proportion of 25% of flies below it')
```

```
0.0051721486014990265 or 0.5172148601499027% of flies have a thorax length less than 0.6mm

0.09991232866619182 or 9.991232866619182% of flies have a thorax length greater than 0.9mm

0.8949155227323091 or 89.49155227323091% of flies have a thorax length between 0.6mm and 0.9mm

0.7473897994847056mm thorax length gives a proportion of 25% of flies below it
```

4. a) Time is the explanatory variable and distance is the response variable. The pattern shown is linear and positively associated (the greater the time, the greater the distance as well).

In [25]:
```
# Question 4a,b,c

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile
from scipy.stats import norm
from scipy import stats


gorillas_csv = pandas.read_csv('gorillas.csv')

sns.scatterplot(x='Time',y='Distance', data = gorillas_csv)
plt.title('Infection of Gorillas - Time vs Distance')

r = stats.pearsonr(gorillas_csv['Distance'], gorillas_csv['Time'])[0]

print(str(r) + ' is the Pearson correlation coefficient r between distance and time')

gorillas_csv['Time'] = gorillas_csv['Time'].div(7)

r7 = stats.pearsonr(gorillas_csv['Distance'], gorillas_csv['Time'])[0]

print(str(r7) + ' is the Pearson correlation coefficient r between distance and time (in weeks)')
```

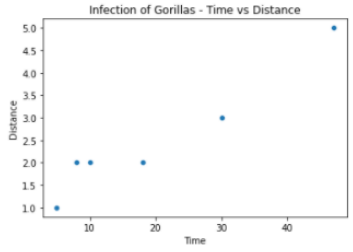
```
0.962338138785128 is the Pearson correlation coefficient r between distance and time
0.962338138785128 is the Pearson correlation coefficient r between distance and time (in weeks)
```



b)

In [25]:
```
# Question 4a,b,c

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile
from scipy.stats import norm
from scipy import stats


gorillas_csv = pandas.read_csv('gorillas.csv')

sns.scatterplot(x='Time',y='Distance', data = gorillas_csv)
plt.title('Infection of Gorillas - Time vs Distance')

r = stats.pearsonr(gorillas_csv['Distance'], gorillas_csv['Time'])[0]

print(str(r) + ' is the Pearson correlation coefficient r between distance and time')

gorillas_csv['Time'] = gorillas_csv['Time'].div(7)

r7 = stats.pearsonr(gorillas_csv['Distance'], gorillas_csv['Time'])[0]

print(str(r7) + ' is the Pearson correlation coefficient r between distance and time (in weeks)')
```

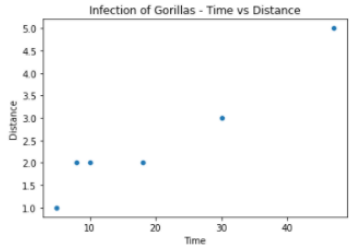
```
0.962338138785128 is the Pearson correlation coefficient r between distance and time
0.962338138785128 is the Pearson correlation coefficient r between distance and time (in weeks)
```



c) No, the correlation r did not change because the correlation is proportional: it depends on the ratio between x and y values. If the ratio between each x and y value remains proportional to the original data, the r value will not change.

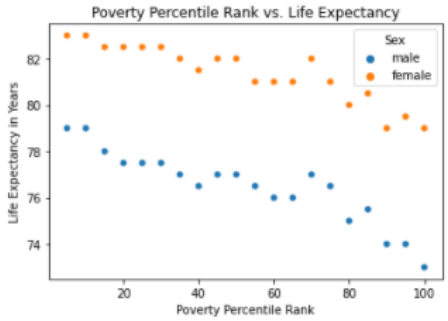
In [25]:
```
# Question 4a,b,c

import numpy
import pandas
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import percentile
from scipy.stats import norm
from scipy import stats


gorillas_csv = pandas.read_csv('gorillas.csv')

sns.scatterplot(x='Time',y='Distance', data = gorillas_csv)
plt.title('Infection of Gorillas - Time vs Distance')

r = stats.pearsonr(gorillas_csv['Distance'], gorillas_csv['Time'])[0]

print(str(r) + ' is the Pearson correlation coefficient r between distance and time')

gorillas_csv['Time'] = gorillas_csv['Time'].div(7)

r7 = stats.pearsonr(gorillas_csv['Distance'], gorillas_csv['Time'])[0]

print(str(r7) + ' is the Pearson correlation coefficient r between distance and time (in weeks)')
```

```
0.962338138785128 is the Pearson correlation coefficient r between distance and time
0.962338138785128 is the Pearson correlation coefficient r between distance and time (in weeks)
```



5. a)

In [26]:
```
# Question 5a,c,d

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

life_csv = pandas.read_csv("lifeexp.csv")

sns.scatterplot(x = "PovertyPercentileRank", y = "LifeExpectancy", hue = 'Sex', data = life_csv)
plt.title("Poverty Percentile Rank vs. Life Expectancy")
plt.xlabel("Poverty Percentile Rank")
plt.ylabel("Life Expectancy in Years")
plt.show()

r = stats.pearsonr(life_csv['LifeExpectancy'], life_csv['PovertyPercentileRank'])[0]
print(str(r) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for me

maledf = life_csv.loc[life_csv['Sex']=='male']

femaledf = life_csv.loc[life_csv['Sex']=='female']

r2 = stats.pearsonr(maledf['LifeExpectancy'], maledf['PovertyPercentileRank'])[0]
print(str(r2) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m

r3 = stats.pearsonr(femaledf['LifeExpectancy'], femaledf['PovertyPercentileRank'])[0]
print(str(r3) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for w
```

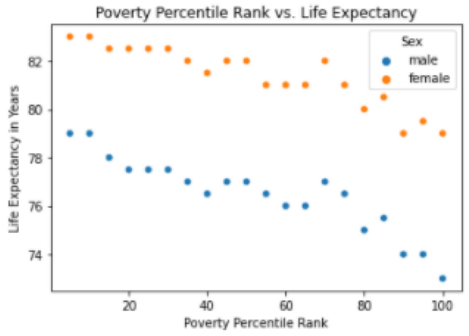



```
-0.4533582802983435 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m
en and women altogether
-0.9239043960699259 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m
en
-0.9266657455577724 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for w
omen
```

b) The plot shows that the relationship between the poverty percentile rank and life expectancy is negatively associated which means that the higher the poverty percentile rank, the lower the life expectancy, for both genders. Sex does not play a factor in the relationship (slope) between the poverty percentile rank and life expectancy, though it does appear that the regression line for women does have a higher y-intercept (longer life expectancy).

c) It shows that the relationship is negatively associated and is moderately strong because -0.45 is almost halfway between 0 and -1.

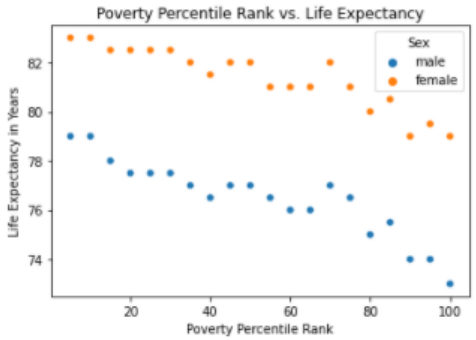
In [26]:
```
# Question 5a,c,d

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

life_csv = pandas.read_csv("lifeexp.csv")

sns.scatterplot(x = "PovertyPercentileRank", y = "LifeExpectancy", hue = 'Sex', data = life_csv)
plt.title("Poverty Percentile Rank vs. Life Expectancy")
plt.xlabel("Poverty Percentile Rank")
plt.ylabel("Life Expectancy in Years")
plt.show()

r = stats.pearsonr(life_csv['LifeExpectancy'], life_csv['PovertyPercentileRank'])[0]
print(str(r) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for me

maledf = life_csv.loc[life_csv['Sex']=='male']

femaledf = life_csv.loc[life_csv['Sex']=='female']

r2 = stats.pearsonr(maledf['LifeExpectancy'], maledf['PovertyPercentileRank'])[0]
print(str(r2) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m

r3 = stats.pearsonr(femaledf['LifeExpectancy'], femaledf['PovertyPercentileRank'])[0]
print(str(r3) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for w
```




```
-0.4533582802983435 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m
en and women altogether
-0.9239043960699259 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m
en
-0.9266657455577724 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for w
omen
```

d) Both the **r** value for males and females indicate a much stronger relationship than when they are grouped together. For males, the r value is -0.924 and for females, the r value is -0.927. These values are more than double the r value when both males and females are grouped together.

In [26]:
```
# Question 5a,c,d

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

life_csv = pandas.read_csv("lifeexp.csv")

sns.scatterplot(x = "PovertyPercentileRank", y = "LifeExpectancy", hue = 'Sex', data = life_csv)
plt.title("Poverty Percentile Rank vs. Life Expectancy")
plt.xlabel("Poverty Percentile Rank")
plt.ylabel("Life Expectancy in Years")
plt.show()

r = stats.pearsonr(life_csv['LifeExpectancy'], life_csv['PovertyPercentileRank'])[0]
print(str(r) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for me

maledf = life_csv.loc[life_csv['Sex']=='male']

femaledf = life_csv.loc[life_csv['Sex']=='female']

r2 = stats.pearsonr(maledf['LifeExpectancy'], maledf['PovertyPercentileRank'])[0]
print(str(r2) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m

r3 = stats.pearsonr(femaledf['LifeExpectancy'], femaledf['PovertyPercentileRank'])[0]
print(str(r3) + ' is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for w
```

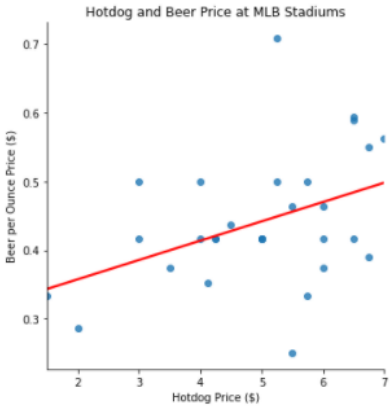



```
-0.4533582802983435 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m
en and women altogether
-0.9239043960699259 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for m
en
-0.9266657455577724 is the Pearson correlation coefficient r between life expectancies and poverty level percentile ranks for w
omen
```

e) Yes, it is important to consider the effect of sex when analyzing the relationship between life expectancy and poverty because when one considers them both together as a whole, they get a less accurate representation of the correlation (not as strong). Though the correlation (slope) may be the same for both genders, when they are grouped together, the correlation is affected greatly because it fails to consider the individual y-intercepts (regression line).

6. a) The plot says that as hotdog prices increase, beer prices increase as well. There is a positive correlation between the two item prices.

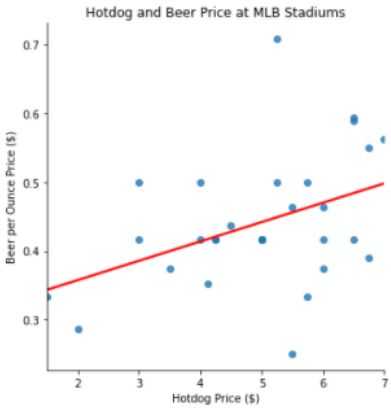
In [28]:
```
# Question 6a,b,e

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

beer_csv = pandas.read_csv('beer.csv')

sns.lmplot(x = "hotdog", y="beer",data = beer_csv, ci=None, line_kws={'color': 'red'})
plt.xlabel("Hotdog Price ($)")
plt.ylabel("Beer per Ounce Price ($)")
plt.title("Hotdog and Beer Price at MLB Stadiums")
plt.show()

model = stats.linregress(x = beer_csv['hotdog'], y = beer_csv['beer'])

print("y = " + str(model.slope) + "x + " + str(model.intercept) + " is the least-squares regression line")

X = pandas.DataFrame({"hotdog":np.linspace(0, 9, 45)})

y = model.intercept + model.slope * X
# x = hotdog dollars, y = ounce of beer per dollar

r = model.rvalue
r2 = (model.rvalue**2)
print(str(r2) + ' or ' + str(r2*100) + '% is r squared')
```




```
y = 0.028102417285971203x + 0.30135248690477046 is the least-squares regression line
0.16906416197532598 or 16.906416197532597% is r squared
```

b)

In [28]:
```
# Question 6a,b,e

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

beer_csv = pandas.read_csv('beer.csv')

sns.lmplot(x = "hotdog", y="beer",data = beer_csv, ci=None, line_kws={'color': 'red'})
plt.xlabel("Hotdog Price ($)")
plt.ylabel("Beer per Ounce Price ($)")
plt.title("Hotdog and Beer Price at MLB Stadiums")
plt.show()

model = stats.linregress(x = beer_csv['hotdog'], y = beer_csv['beer'])

print("y = " + str(model.slope) + "x + " + str(model.intercept) + " is the least-squares regression line")

X = pandas.DataFrame({"hotdog":np.linspace(0, 9, 45)})

y = model.intercept + model.slope * X
# x = hotdog dollars, y = ounce of beer per dollar

r = model.rvalue
r2 = (model.rvalue**2)
print(str(r2) + ' or ' + str(r2*100) + '% is r squared')
```




```
y = 0.028102417285971203x + 0.30135248690477046 is the least-squares regression line
0.16906416197532598 or 16.906416197532597% is r squared
```

c) The slope says that for every dollar for a hotdog, a beer is $0.0281 (per ounce) on average.

d) 0.30 = 0.028102417285971203x + 0.30135248690477046

-0.001352486 = 0.028102417285971203x

x = -0.048127066

The predicted price of a hotdog when one ounce of beer costs 0.30 dollars is -0.048127066 dollars. We can consider this to be 0 dollars (free!).

0.60 = 0.028102417285971203x + 0.30135248690477046

0.298647513 = 0.028102417285971203x

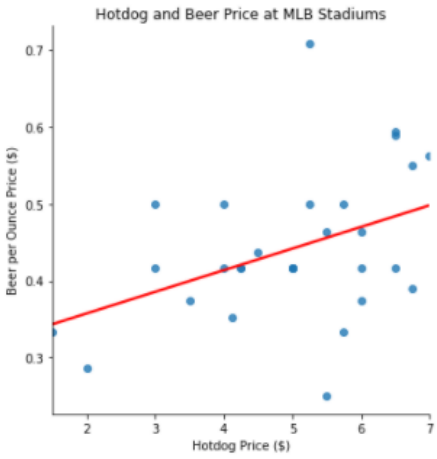
x = 10.62711119

The predicted price of a hotdog when one ounce of beer costs 0.60 dollars is 10.62711119 dollars. We can consider this to be 10.63 dollars.

The prices can vary greatly and can even go in the negatives (when domain is not restricted).

e) 16.9% (r squared) of the variation in price of a hotdog is explained by the fitted regression line. This number (16.9%) says that there is a weak relationship between the price of a hotdog and price of bear as only 16.9% is explained by the regression line. This means that 83.1% is due to random/unexplained factors.

In [28]:
```
# Question 6a,b,e

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

beer_csv = pandas.read_csv('beer.csv')

sns.lmplot(x = "hotdog", y="beer",data = beer_csv, ci=None, line_kws={'color': 'red'})
plt.xlabel("Hotdog Price ($)")
plt.ylabel("Beer per Ounce Price ($)")
plt.title("Hotdog and Beer Price at MLB Stadiums")
plt.show()

model = stats.linregress(x = beer_csv['hotdog'], y = beer_csv['beer'])

print("y = " + str(model.slope) + "x + " + str(model.intercept) + " is the least-squares regression line")

X = pandas.DataFrame({"hotdog":np.linspace(0, 9, 45)})

y = model.intercept + model.slope * X
# x = hotdog dollars, y = ounce of beer per dollar

r = model.rvalue
r2 = (model.rvalue**2)
print(str(r2) + ' or ' + str(r2*100) + '% is r squared')
```

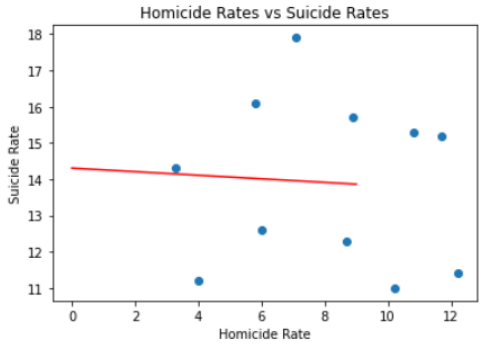



```
y = 0.028102417285971203x + 0.30135248690477046 is the least-squares regression line
0.16906416197532598 or 16.906416197532597% is r squared
```

7. No, this does not mean that drinking more diet sodas results in more weight gain. There is most certainly a lurking variable that is in effect in this study. The lurking variable could be regular sodas or any other junk food. It is likely that the more regular sodas one drinks, the more diet sodas as well. Regular soda is possibly the real explanatory variable in this scenario because one can't gain weight from diet sodas.

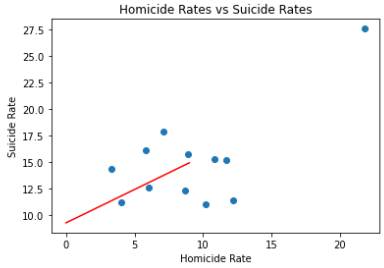
8. a) The scatterplot shows very little (negatively correlated) to no pattern between homicide and suicide rates within the various counties. Because of this, the suicide rate can't really be predicted from the homicide rate.

In [33]:
```
# Question 8

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

homicide_csv = pandas.read_csv('homicide.csv')

X = pandas.DataFrame({"Homicide":np.linspace(0, 9, 45)})

model = stats.linregress(x = homicide_csv['Homicide'], y = homicide_csv['Suicide'])
y_pred = model.intercept + model.slope * X

plt.scatter(homicide_csv['Homicide'], homicide_csv['Suicide'])
plt.plot(X, y_pred, color = 'red', label = 'fitted line')
plt.title("Homicide Rates vs Suicide Rates")
plt.xlabel("Homicide Rate")
plt.ylabel("Suicide Rate")
plt.show()
```

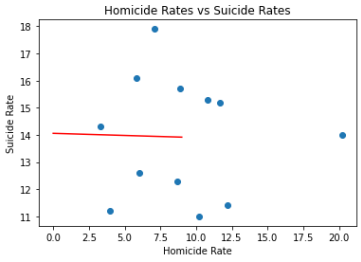


b) Point A is an outlier that appears to be above (higher y value) and to the right (higher x value) of the main set of data. Point B is an outlier that appears to be only to the right (higher x value) of the main set of data. Though it may appear that Point B is an outlier because of the large gap between itself and the main set of data, it seems to follow the original trend still.

In [34]:
```
# Question 8

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

homicide_csv = pandas.read_csv('homicide.csv')

p1 = pandas.DataFrame({"Homicide":[21.8], "Suicide":[27.6]})

homicide_csv = homicide_csv.append(p1, ignore_index=True)

X = pandas.DataFrame({"Homicide":np.linspace(0, 9, 45)})

model = stats.linregress(x = homicide_csv['Homicide'], y = homicide_csv['Suicide'])
y_pred = model.intercept + model.slope * X

plt.scatter(homicide_csv['Homicide'], homicide_csv['Suicide'])
plt.plot(X, y_pred, color = 'red', label = 'fitted line')
plt.title("Homicide Rates vs Suicide Rates")
plt.xlabel("Homicide Rate")
plt.ylabel("Suicide Rate")
plt.show()
```

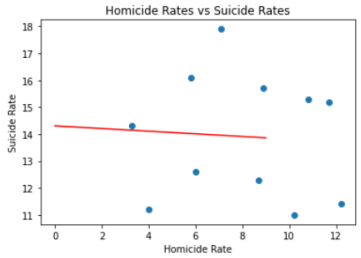


In [35]:
```
# Question 8

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

homicide_csv = pandas.read_csv('homicide.csv')

p2 = pandas.DataFrame({"Homicide":[20.2], "Suicide":[14.0]})
homicide_csv = homicide_csv.append(p2, ignore_index=True)

X = pandas.DataFrame({"Homicide":np.linspace(0, 9, 45)})

model = stats.linregress(x = homicide_csv['Homicide'], y = homicide_csv['Suicide'])
y_pred = model.intercept + model.slope * X

plt.scatter(homicide_csv['Homicide'], homicide_csv['Suicide'])
plt.plot(X, y_pred, color = 'red', label = 'fitted line')
plt.title("Homicide Rates vs Suicide Rates")
plt.xlabel("Homicide Rate")
plt.ylabel("Suicide Rate")
plt.show()
```

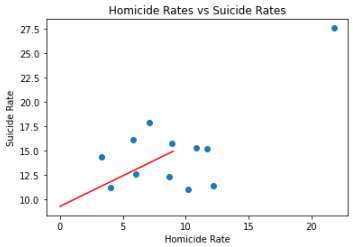


c) Point A was much more influential than Point B as it changed the slope of the least-squares regression line much more than did Point B. Point A moved the line to make it more steep (bigger slope) because Point A had a very large y-value relative to the set of data. Point B barely moved the line because its data seemed to fit with the original data (same ratio between x and y, x:y).

In [33]:
```
# Question 8

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

homicide_csv = pandas.read_csv('homicide.csv')

X = pandas.DataFrame({"Homicide":np.linspace(0, 9, 45)})

model = stats.linregress(x = homicide_csv['Homicide'], y = homicide_csv['Suicide'])
y_pred = model.intercept + model.slope * X

plt.scatter(homicide_csv['Homicide'], homicide_csv['Suicide'])
plt.plot(X, y_pred, color = 'red', label = 'fitted line')
plt.title("Homicide Rates vs Suicide Rates")
plt.xlabel("Homicide Rate")
plt.ylabel("Suicide Rate")
plt.show()
```



In [34]:
```
# Question 8

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

homicide_csv = pandas.read_csv('homicide.csv')

p1 = pandas.DataFrame({"Homicide":[21.8], "Suicide":[27.6]})

homicide_csv = homicide_csv.append(p1, ignore_index=True)

X = pandas.DataFrame({"Homicide":np.linspace(0, 9, 45)})

model = stats.linregress(x = homicide_csv['Homicide'], y = homicide_csv['Suicide'])
y_pred = model.intercept + model.slope * X

plt.scatter(homicide_csv['Homicide'], homicide_csv['Suicide'])
plt.plot(X, y_pred, color = 'red', label = 'fitted line')
plt.title("Homicide Rates vs Suicide Rates")
plt.xlabel("Homicide Rate")
plt.ylabel("Suicide Rate")
plt.show()
```



In [35]:
```
# Question 8

import numpy as np
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

homicide_csv = pandas.read_csv('homicide.csv')

p2 = pandas.DataFrame({"Homicide":[20.2], "Suicide":[14.0]})
homicide_csv = homicide_csv.append(p2, ignore_index=True)

X = pandas.DataFrame({"Homicide":np.linspace(0, 9, 45)})

model = stats.linregress(x = homicide_csv['Homicide'], y = homicide_csv['Suicide'])
y_pred = model.intercept + model.slope * X

plt.scatter(homicide_csv['Homicide'], homicide_csv['Suicide'])
plt.plot(X, y_pred, color = 'red', label = 'fitted line')
plt.title("Homicide Rates vs Suicide Rates")
plt.xlabel("Homicide Rate")
plt.ylabel("Suicide Rate")
plt.show()
```