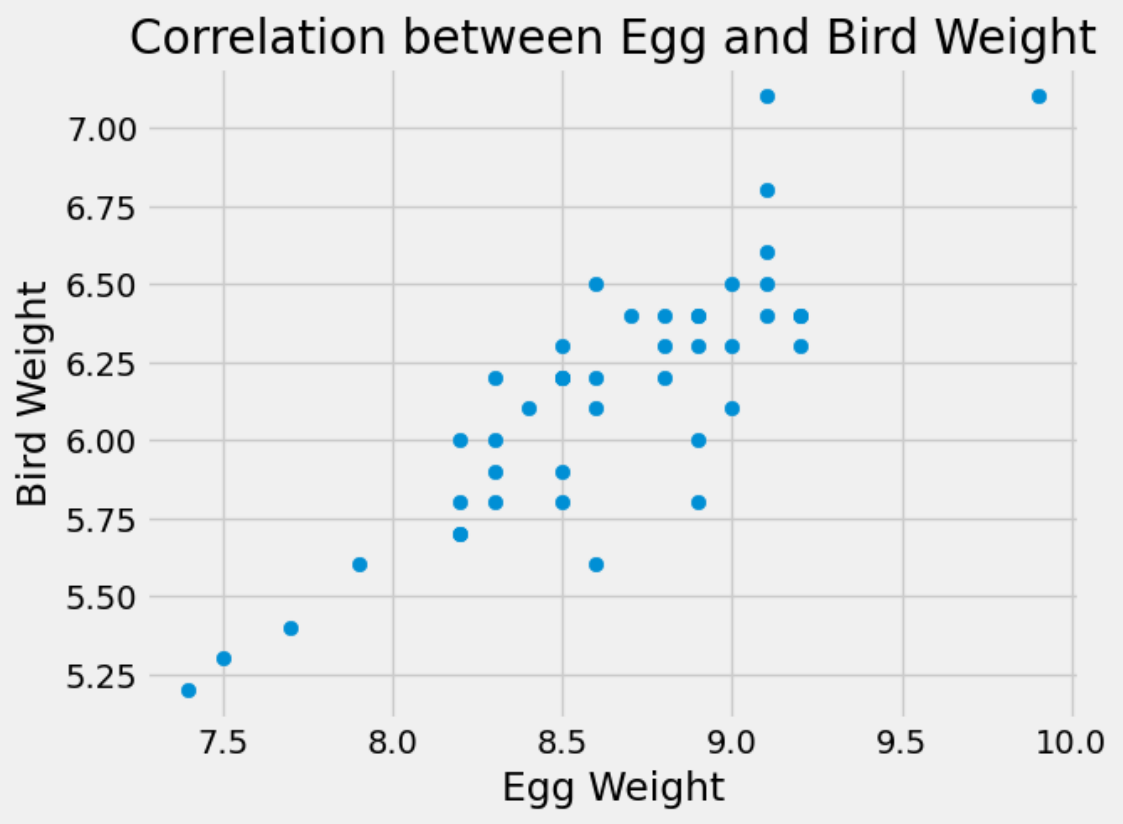
Question 1.1.

a). Create a scatter plot of the egg weights (on the x-axis) vs the bird weights (on the y-axis). Label your axes and give your plot a title.

b). Based only on your plot, make a guess as to what the correlation is between these two variables and assign it to the variable `corr_guess` (don't do any actual calculations yet, just guess based on your visual inspection of the plot).

In [4]:
```
# a)
plt.scatter(birds['Egg Weight'], birds['Bird Weight'])
plt.xlabel('Egg Weight')
plt.ylabel('Bird Weight')
plt.title('Correlation between Egg and Bird Weight')
plt.show()
# Your code for part (a) above this line
```



In [5]:
```
corr_guess = 0.8
```

Question 1.4

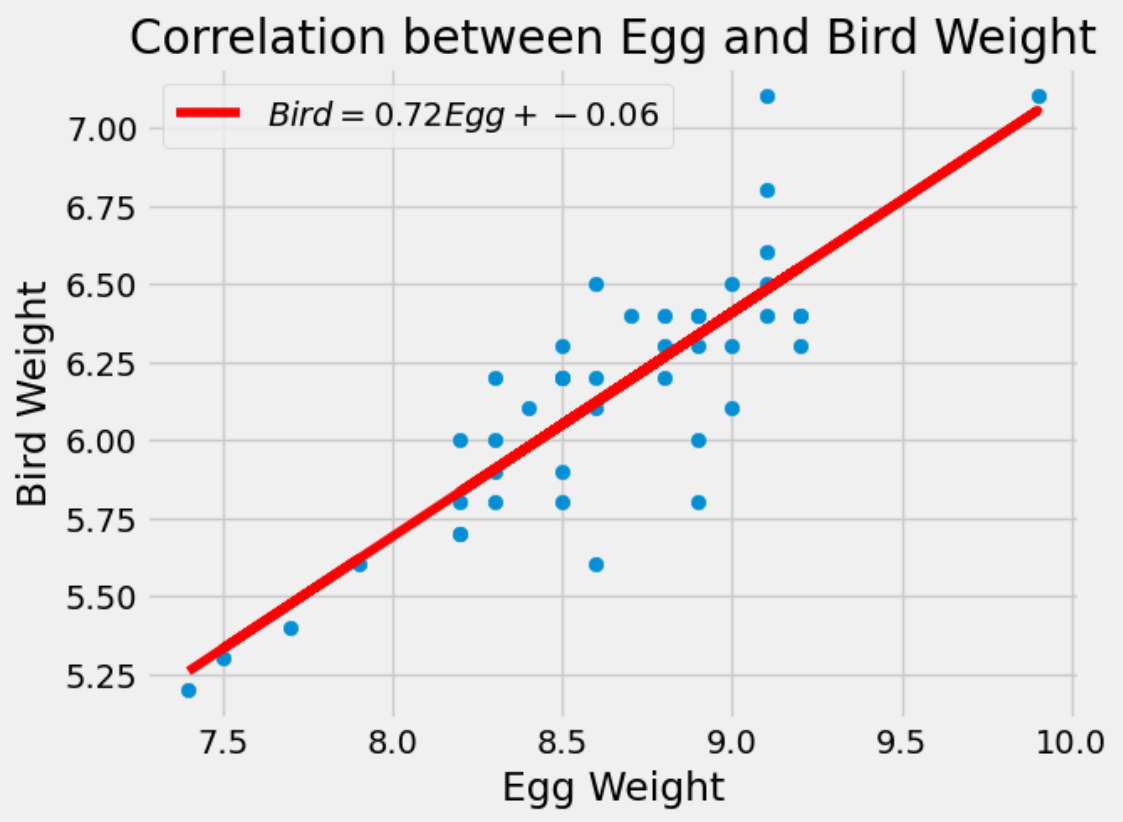
Part a).

- Run `fit_line` on the `birds` table.
- Then create a scatterplot of the birds data with an overlaid plot of the least squares linear regression line (using the output of your `fit_line` function).
- For credit on this problem you must use the output of your `fit_line` function to create the line yourself, to demonstrate you understand how this line is created.
- Label your axes and create a label for the line on the plot that gives the equation of the line. Tip: including the following code in your plt.plot function will add a label with the equation of the best fit line: `label =r'$Bird = {0:.2f}Egg{1:.2f}$'.format(slope, intercept)`

Part b). Based on the slope from your least squares regression line model, we see there is a positive linear association in this sample of data between Snowy Plover egg weight and bird weight. Can we conclude from this that there is also a positive linear association between egg weight and bird weight in the population of Snowy Plover birds? Why or why not? Explain your reasoning.

Part b answer cell: *Put your answer to part b in this cell...* The positive linear association a positive slope between egg weight and bird weight. In order to conclude that there is also a positive linear association between the egg weight and bird weight of Snowy Plover birds, we need to know its statistical significance. We will need a hypothesis test and confidence intervals to see whether the observed is in a broader population or just random sampling variability in the data. Therefore, we can't conclude that there is a positive linear association in the population just based off of the correlation and best fit line graph of a positive slope.

```
In [10]: birdLine = fit_line(birds, 'Egg Weight', 'Bird Weight')
         plt.scatter(birds['Egg Weight'], birds['Bird Weight'])
         plt.plot(birds['Egg Weight'], birdLine[0] * birds['Egg Weight'] + birdLine[1], color='red',
                  label=r'$Bird = {0:.2f}Egg + {1:.2f}$'.format(birdLine[0], birdLine[1]))
         plt.xlabel('Egg Weight')
         plt.ylabel('Bird Weight')
         plt.title('Correlation between Egg and Bird Weight')
         plt.legend()
         plt.show()# Your code for part a above this line
```


Correlation between Egg and Bird Weight
$Bird = 0.72Egg + -0.06$
Bird Weight
Egg Weight
7.00
6.75
6.50
6.25
6.00
5.75
5.50
5.25
7.5
8.0
8.5
9.0
9.5
10.0

Question 2.2.

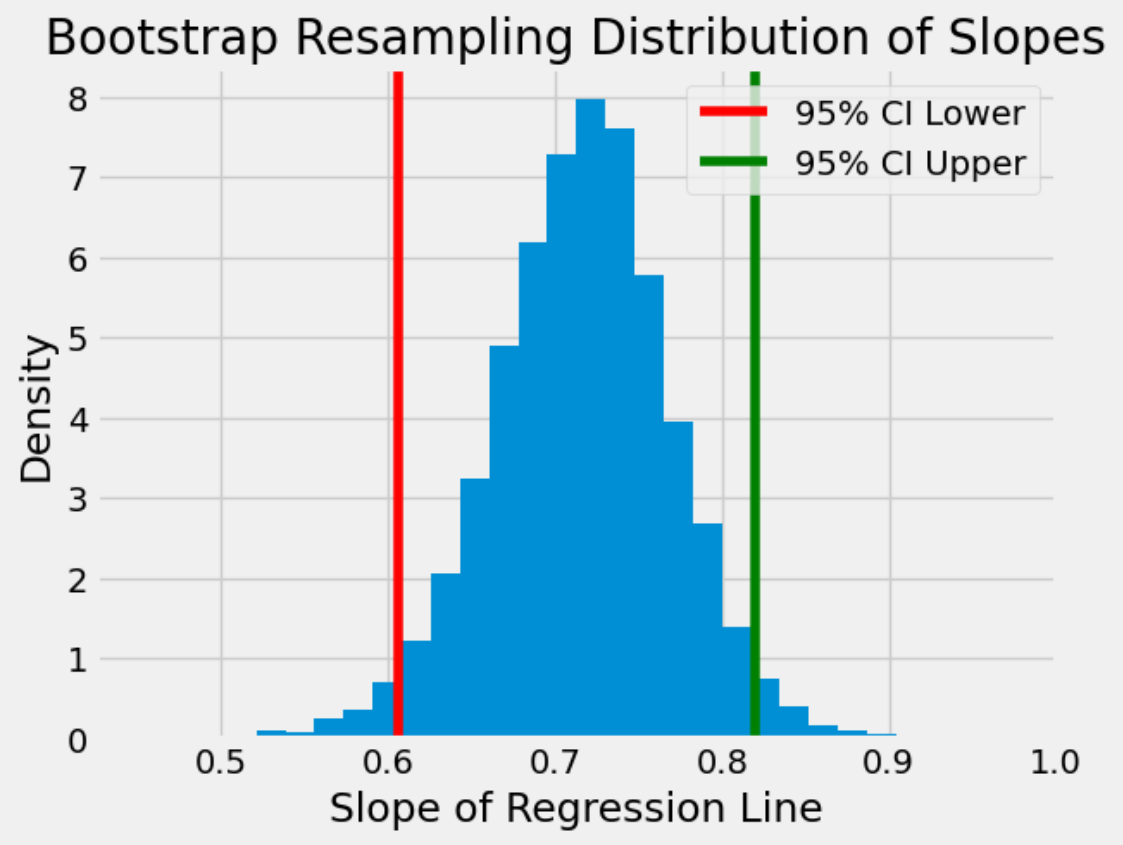
a). Create a *numpy array* called `resampled_slopes` that contains the slope of the best fit line for 10,000 bootstrap resamples of `birds`. (Hint, use your function `fit_line` from question 1).

b). Then create a 95% Confidence Interval for the true value of the slopes and assign the lower and upper values to the variables `CI_lower` and `CI_upper` respectively.

c). Create a plot with a histogram of the density distribution of these slopes AND the confidence interval overlaid on the bottom of the distribution (similar to histograms you made in HW 10).

```
In [13]: resampled_slopes = np.array([fit_line(birds.sample(frac = 1, replace = True),
                                               'Egg Weight', 'Bird Weight')[0]
                                       for i in range (10000)])

         # your code for part (a) above

         CI_lower = np.percentile(resampled_slopes, 2.5)
         CI_upper = np.percentile(resampled_slopes, 97.5)

         print("95% confidence interval for slope: [{:g}, {:g}]".format(CI_lower, CI_upper))



         # your code for part (c) above
         plt.hist(resampled_slopes, bins=30, density=True,)
         plt.axvline(x=CI_lower, color='red', label='95% CI Lower')
         plt.axvline(x=CI_upper, color='green', label='95% CI Upper')
         plt.title('Bootstrap Resampling Distribution of Slopes')
         plt.xlabel('Slope of Regression Line')
         plt.ylabel('Density')
         plt.legend()
         plt.show()
```

```
95% confidence interval for slope: [0.605807, 0.819466]
```



In [14]: grader.check("q2_2")

Out[14]: q2_2 results: All test cases passed!

Question 2.3. Based on your confidence interval, would you accept or reject the null hypothesis that the true slope is 0? Explain your reasoning. What p-value cutoff are you using?

I would reject the null hypothesis that the true slope is 0, as CI of 95% is [0.605303, 0.819618] which isn't within the interval. We are using a 0.05 p-value for 95% confidence interval. Thus if the null hypothesis is true, we will see that 0 will be within the confidence interval 95% of the time. However this isn't the case, and we will reject the null hypothesis, as the CI has a range of values that doesn't include 0 so we see that we have confidence that the true slope is different from 0.

Question 2.7 Define the function `bootstrap_lines`. It takes in four arguments: 1. `df`: a dataframe like `birds` 2. `x_col`: the name of our x-column within the input `tbl` 3. `y_col`: the name of our y-column within the input `tbl` 4. `num_bootstraps`: an integer, a number of bootstraps to run.

It returns a *dataframe* with one row for each bootstrap resample and the following two columns: 1. `Slope`: the bootstrapped slopes 2. `Intercept`: the corresponding bootstrapped intercepts

(Hint, use your function from the previous part of this question)

Then call this function 10,000 times using the bird data.

```
In [22]: def bootstrap_lines(df, x_col, y_col, num_bootstraps):
             resample_line = pd.DataFrame((compute_resampled_line(df, x_col,y_col)
                                           for i in range(num_bootstraps)),
                                          columns = ['Slope', 'Intercept'])
             return resample_line
         regression_lines = bootstrap_lines(birds, "Egg Weight", "Bird Weight", 10000)
         regression_lines.head()
```

```
Out[22]:       Slope  Intercept
         0  0.730301  -0.193024
         1  0.795006  -0.689385
         2  0.697291   0.100338
         3  0.728826  -0.179460
         4  0.720929  -0.085510
```

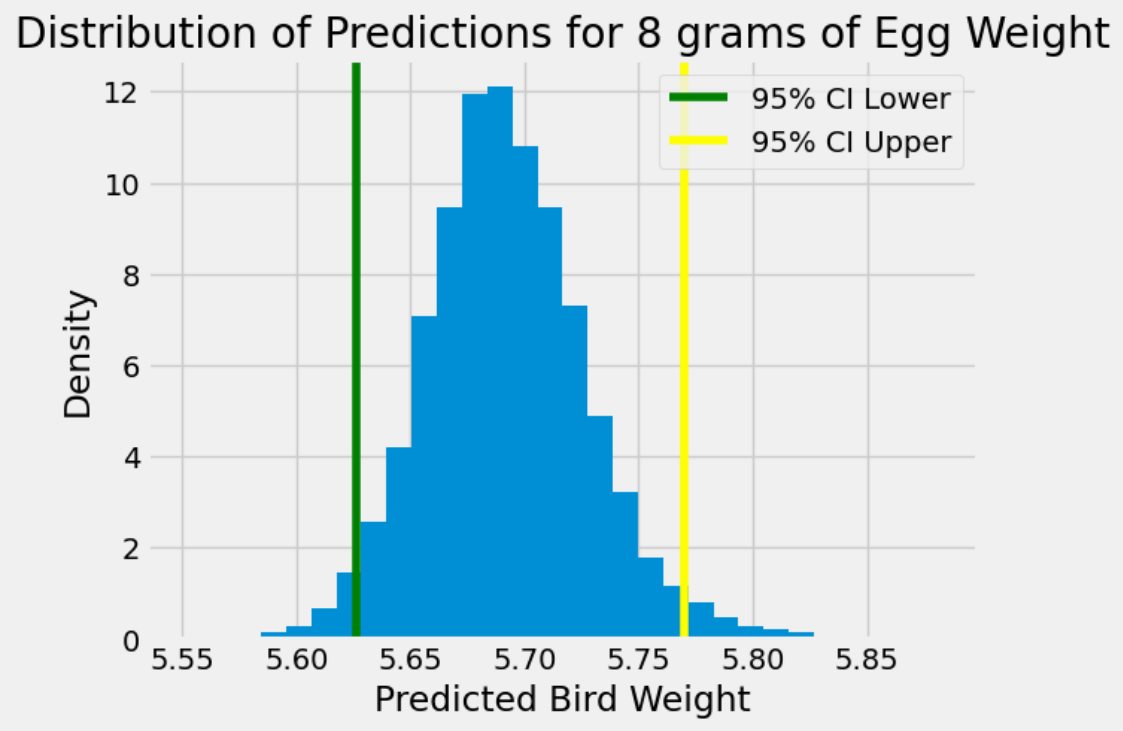
Question 2.8.

a). Create a *numpy array* called `predictions_for_eight` that contains the predicted bird weights based on an egg of weight 8 grams for each regression line in `regression_lines` (from Question 2.7).

b). Then create a 95% Confidence Interval for the true value of the prediction for a weight of 8 grams and assign the lower and upper values to the variables `CI_lower_pred` and `CI_upper_pred` respectively.

c). Create a plot with a histogram of the density distribution of these predictions AND the confidence interval overlaid on the bottom of the distribution. Label your axes on the plot.

```
In [23]: predictions_for_eight = regression_lines['Slope'] * 8 + regression_lines['Intercept']
         # your code for part (a) above

         CI_lower_pred = np.percentile(predictions_for_eight, 2.5)
         CI_upper_pred = np.percentile(predictions_for_eight, 97.5)

         print("95% confidence interval for slope: [{:g}, {:g}]".format(CI_lower_pred, CI_upper_pred))


         plt.hist(predictions_for_eight, bins=30, density=True)
         plt.axvline(x=CI_lower_pred, color='green', label='95% CI Lower')
         plt.axvline(x=CI_upper_pred, color='yellow', label='95% CI Upper')
         plt.title('Distribution of Predictions for 8 grams of Egg Weight')
         plt.xlabel('Predicted Bird Weight')
         plt.ylabel('Density')
         plt.legend()
         plt.show()
         # your code for part (c) above


95% confidence interval for slope: [5.62596, 5.76972]
```




```
In [24]: grader.check("q2_8")
```

```
Out[24]: q2_8 results: All test cases passed!
```

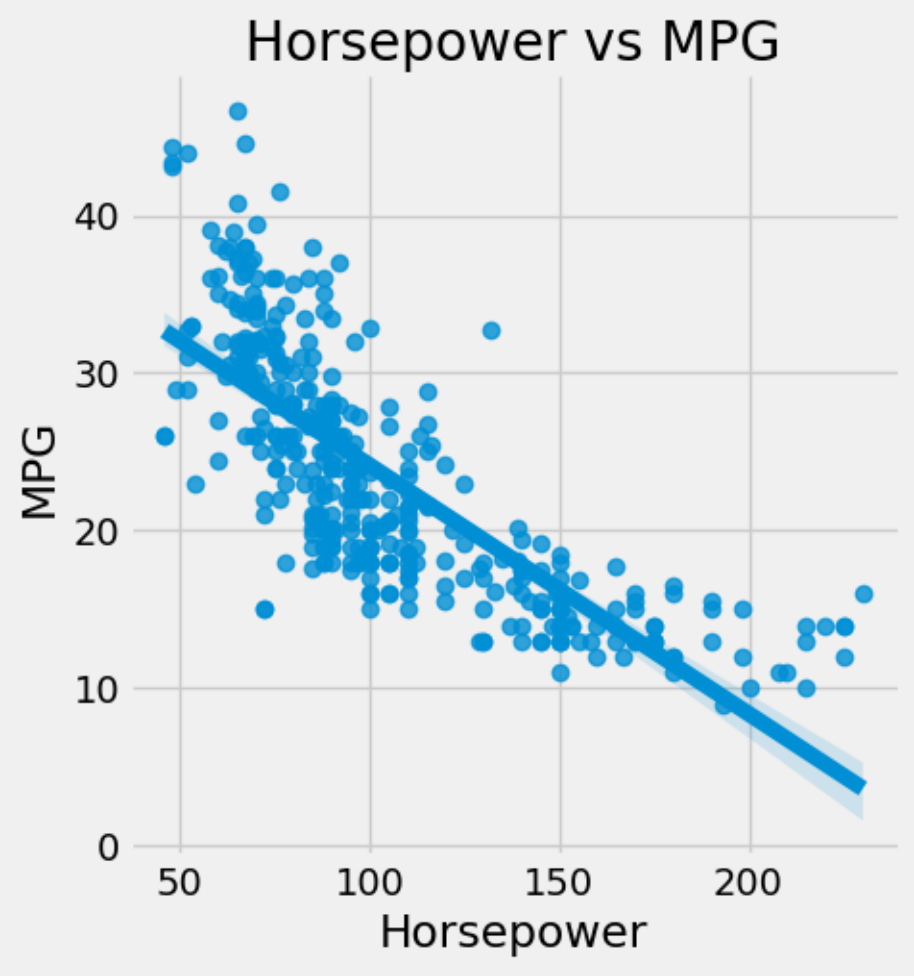
Question 3.1. We'll start by building a simple linear regression model to try and predict `mpg` using `horsepower`.

a). Use your function `fit_line` from question to fit this linear model (i.e. find the slope and intercept for the least squares regression line).

b). Then use seaborn `lmplot` to plot a scatterplot of horsepower (on the x-axis) vs mpg (on the y-axis) with the least squares linear regression line and prediction intervals included on the plot.

```
In [30]: line = fit_line(vehicle_data, 'horsepower', 'mpg')
         slope = line[0]
         intercept = line[1]

         print("Our model slope is ", slope, "and the intercept is", intercept)


         sns.lmplot(data = vehicle_data, x = 'horsepower' , y = 'mpg')
         plt.title('Horsepower vs MPG')
         plt.xlabel('Horsepower')
         plt.ylabel('MPG')
         plt.show()
         # Your code for part b above this line
```

```
Our model slope is  -0.15784473335365343 and the intercept is 39.935861021170446
```


Horsepower vs MPG
MPG
Horsepower
0
10
20
30
40
50
100
150
200

Question 3.3. One way we check model goodness of fit is to analyze the residuals.

For each of the (horsepower, mpg) data points given below, calculate the residual. Give your answer as a **float value**.

Hint: You can use `my_model.predict()` https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegr

Note that the input and the output of this method are numpy arrays, and the test below requires a single float as your answer.

Data point A: (52, 44)

Data point B: (150, 11)

```
In [38]: res_A = ((44 - my_model.predict([[52]])))
         res_B = ((11 - my_model.predict([[150]])))

         print(res_A)
         print(res_B)
```

```
[12.27206511]
[-5.25915102]
```

```
In [39]: grader.check("q3_3")
```

```
Out[39]: q3_3 results: All test cases passed!
```

Question 3.4. Do these residual plots indicate that our linear model is a good model for the data? Why or why not?

The three residual plots do not really indicate that this is a good model for the data.

The SLR plot shows that there is a curve, where is shows that there is a non-linear relationship.

For the residual vs horsepower, we see that it doesn't have a random distribution around 0 and shows a parabola like pattern which shows that this linear model isn't getting the variation in mpg as function of horsepower.

For the residual vs predicted mpg, like residual vs horsepower, it doesn't have a random distribution around 0 and instead as a noticable pattern.

Thus these three models above, isn't a really great lienar model due to patterns and there are limitations of a simple linear regression model.

Question 4.2

Let's start by trying Option 1 in the list above.

a). Add a new column to vehicle_data called `sqrt(hp)` that contains the square root of the horsepower data.

b). Then plot a scatterplot of `mpg` vs `sqrt(hp)` to visually inspect if this transformation makes the data appear more linear than our first model. Label your axes.

```
In [43]: vehicle_data['sqrt(hp)'] = np.sqrt(vehicle_data['horsepower'])
         # your code for part(a) above


         vehicle_data.head()
```

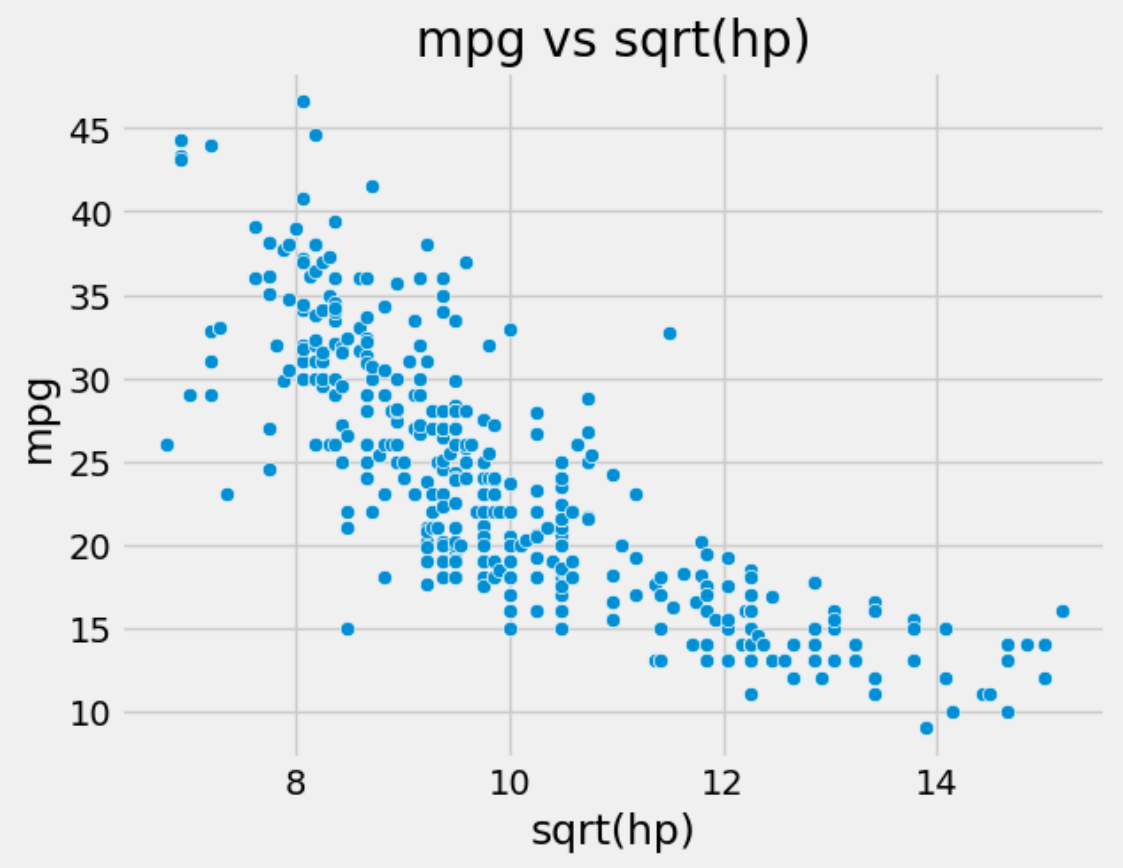
Out[43]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	\
19	26.0	4	97.0	46.0	1835	20.5	
102	26.0	4	97.0	46.0	1950	21.0	
326	43.4	4	90.0	48.0	2335	23.7	
325	44.3	4	90.0	48.0	2085	21.7	
244	43.1	4	90.0	48.0	1985	21.5	

	model_year	origin	name	sqrt(hp)
19	70	europe	volkswagen 1131 deluxe sedan	6.782330
102	73	europe	volkswagen super beetle	6.782330
326	80	europe	vw dasher (diesel)	6.928203
325	80	europe	vw rabbit c (diesel)	6.928203
244	78	europe	volkswagen rabbit custom diesel	6.928203

```
In [44]: sns.scatterplot(data = vehicle_data, x = 'sqrt(hp)', y = 'mpg')
         plt.title('mpg vs sqrt(hp)')
         plt.xlabel('sqrt(hp)')
         plt.ylabel('mpg')
         # your code for part(b) above
```

```
Out[44]: Text(0, 0.5, 'mpg')
```




```
In [45]: grader.check("q4_2")
```

```
Out[45]: q4_2 results: All test cases passed!
```

In the cell below, explain why we use the term "linear" to describe the model above, even though it incorporates a square root function as a feature.

The term linear isn't how its linear to the variables, but how it is linear to the parameters that the model is solving for. The term is used the parameters θ_0 and θ_1 are raised to the power of 1, and through the square root function, it is still linear due to the predictor $\sqrt{(horsepower)}$ and predicted_mpg_hp_sqrt is linear with respect to the coefficients.

Question 4.4.

a). Use `sklearn` to create and fit this new model.

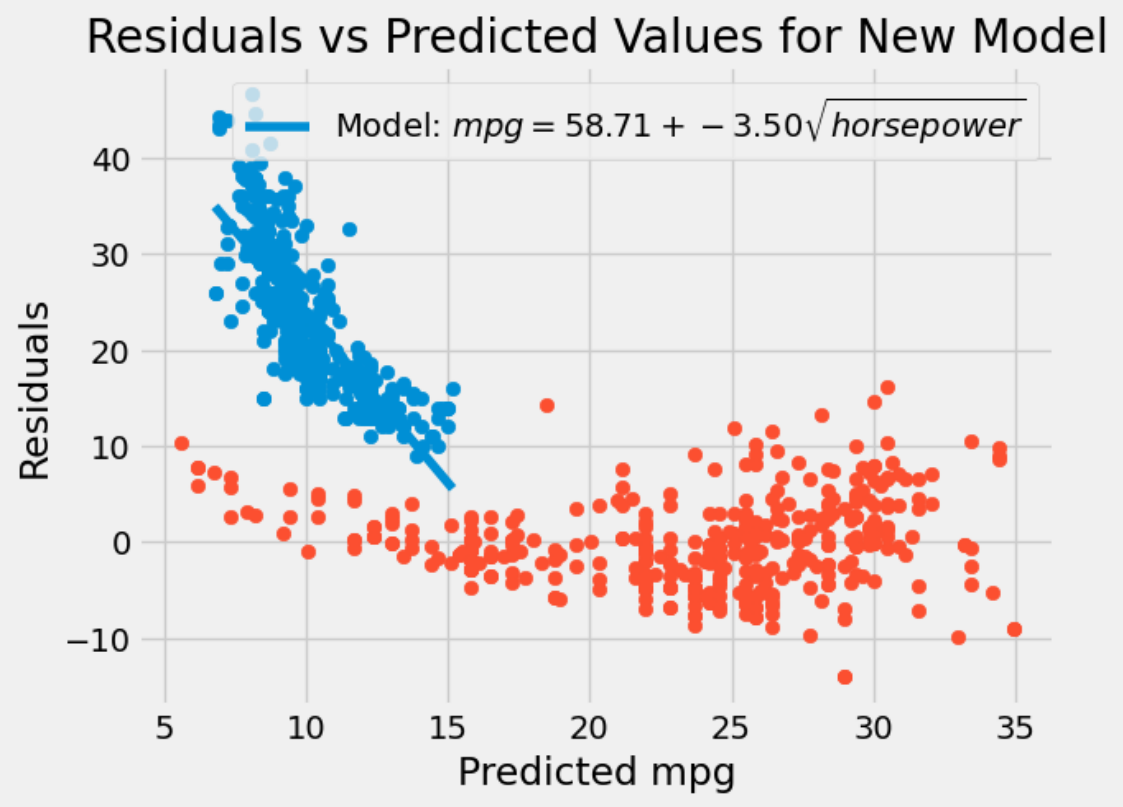
b. Once you've created the new model, set `predicted_mpg_hp_sqrt` to the predicted mpg for the data.

c). Make 2 side-by-side plots:
- A plot of this new model overlaid with a scatterplot of the original data. Include the equation for the new model as a label on your plot (see the plots in question 3 for reference on how to code this). - A plot of the residuals vs the **predicted values of mpg**

d). Calculate the RMSE for this model

```
In [46]: sqrt_hp = vehicle_data[['sqrt(hp)']]
         mpg = vehicle_data['mpg']
         model = lm.LinearRegression()
         model.fit(sqrt_hp,mpg)
         # Your code for parts a) and b) above this line
         predict = model.predict(sqrt_hp)
```

```
In [47]: # Your code for part c  above this line
         plt.scatter(vehicle_data['sqrt(hp)'], mpg)
         plt.plot(vehicle_data['sqrt(hp)'],
                  predict,
                  label=(f'Model: $mpg = {model.intercept_:.2f} + '
                         f'{model.coef_[0]:.2f} \sqrt{{horsepower}}$'))
         plt.xlabel('sqrt(horsepower)')
         plt.ylabel('mpg')
         plt.title('Scatterplot of mpg vs sqrt(horsepower) with New Model')
         plt.legend()

         residual = mpg - predict
         plt.scatter(predict, residual)
         plt.xlabel('Predicted mpg')
         plt.ylabel('Residuals')
         plt.title('Residuals vs Predicted Values for New Model')
         plt.legend()
         plt.show()
```




```
In [48]: RMSE_hp_sqrt = np.sqrt(((residual)**2).mean())
         print("The RMSE of this model is ", RMSE_hp_sqrt)
```

```
The RMSE of this model is  4.6529072608058675
```

```
In [49]: grader.check("q4_4")
```

```
Out[49]: q4_4 results: All test cases passed!
```

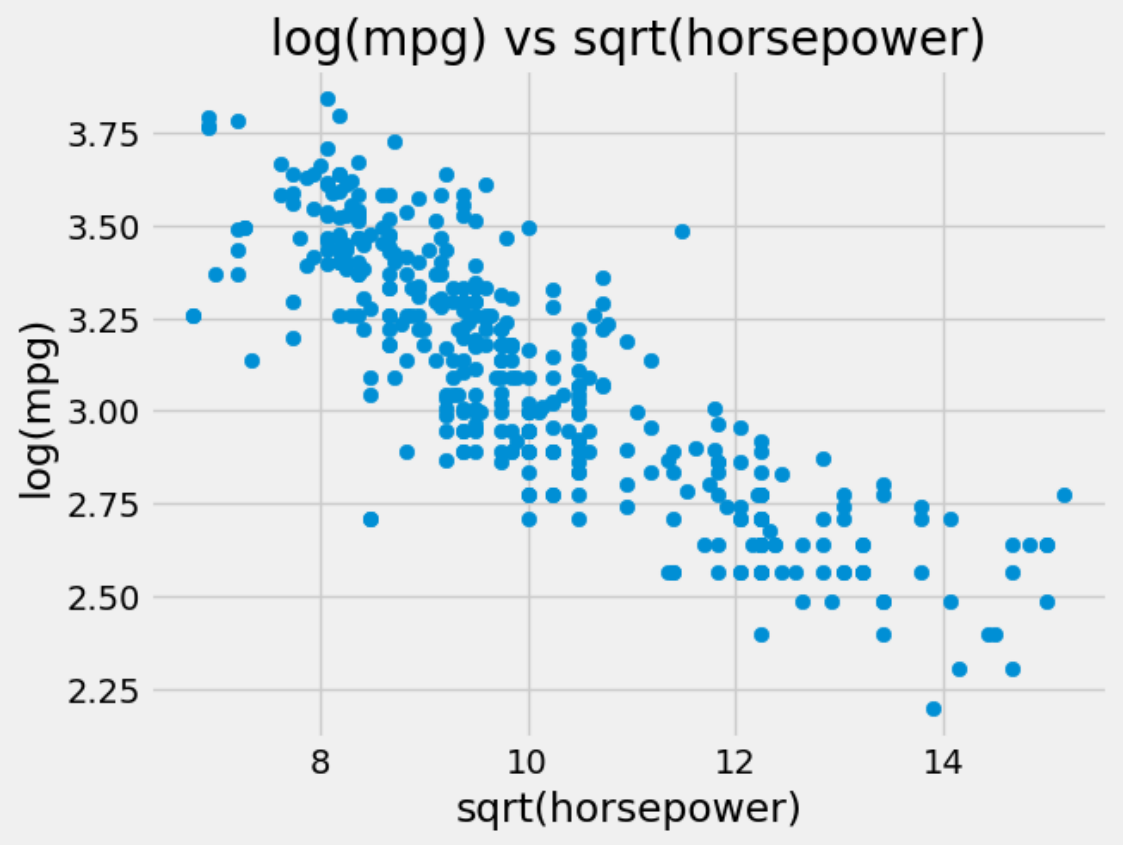
Question 4.5. Analyze the new model compared to the original one. Which RMSE is smaller? Does the residual plot of this new model indicate this model is a good choice? Why or why not?

The new model's RMSE is smaller which does indicate a good choice. In the risdual plot for the new model, the residuals are more evenly distributed around 0. It shows a more clearer pattern, which means that square root transformation helped linearize the relationship between horsepower and mpg. Thus the new model's RMSE is smaller and the model seems to be a good choice over the original.

a). Add a new column to vehicle_data called `log(mpg)` that contains the log of the **mpg** data.

b). Then plot a scatterplot of `log(mpg)` vs `sqrt(hp)` to visually inspect if this transformation makes the data appear more linear than our first and second models. Label your axes.

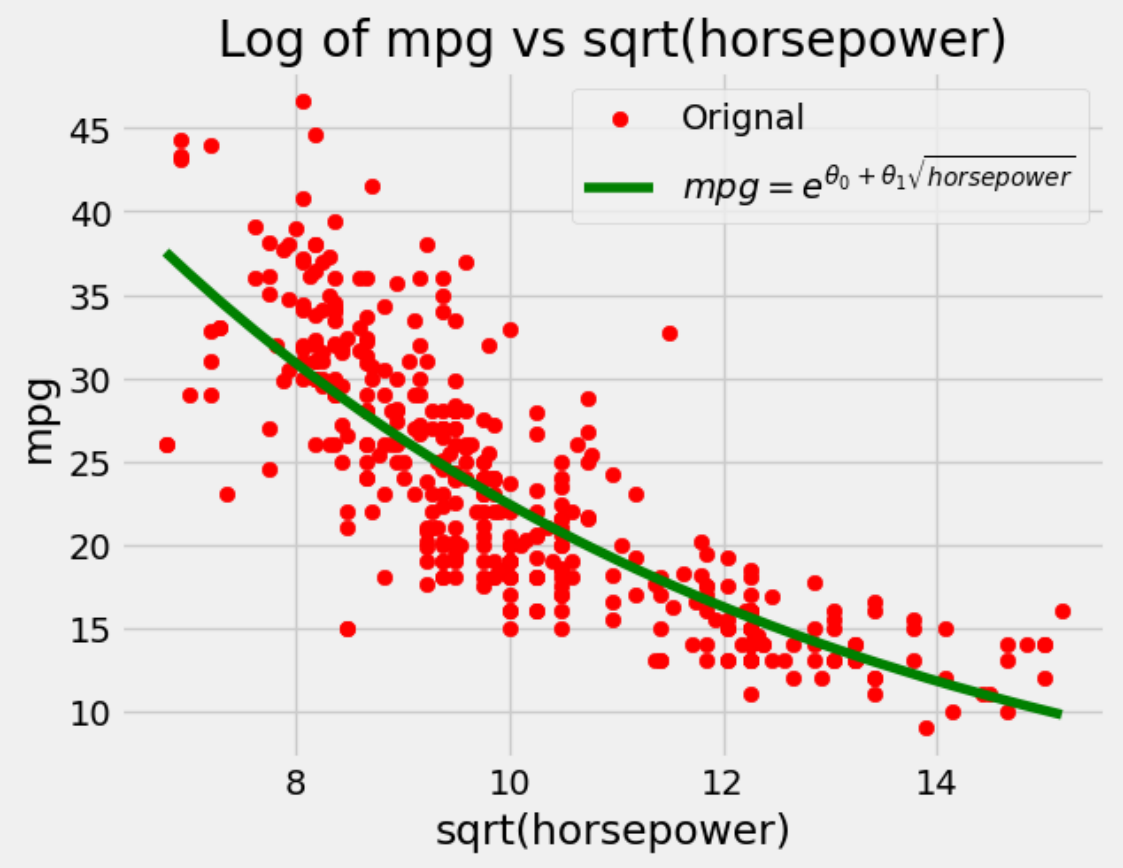
```
In [50]: # a)
         vehicle_data['log(mpg)'] = np.log(vehicle_data['mpg'])
         # b)
         plt.scatter(vehicle_data['sqrt(hp)'], vehicle_data['log(mpg)'])
         plt.xlabel('sqrt(horsepower)')
         plt.ylabel('log(mpg)')
         plt.title('log(mpg) vs sqrt(horsepower)')
         plt.show()
```

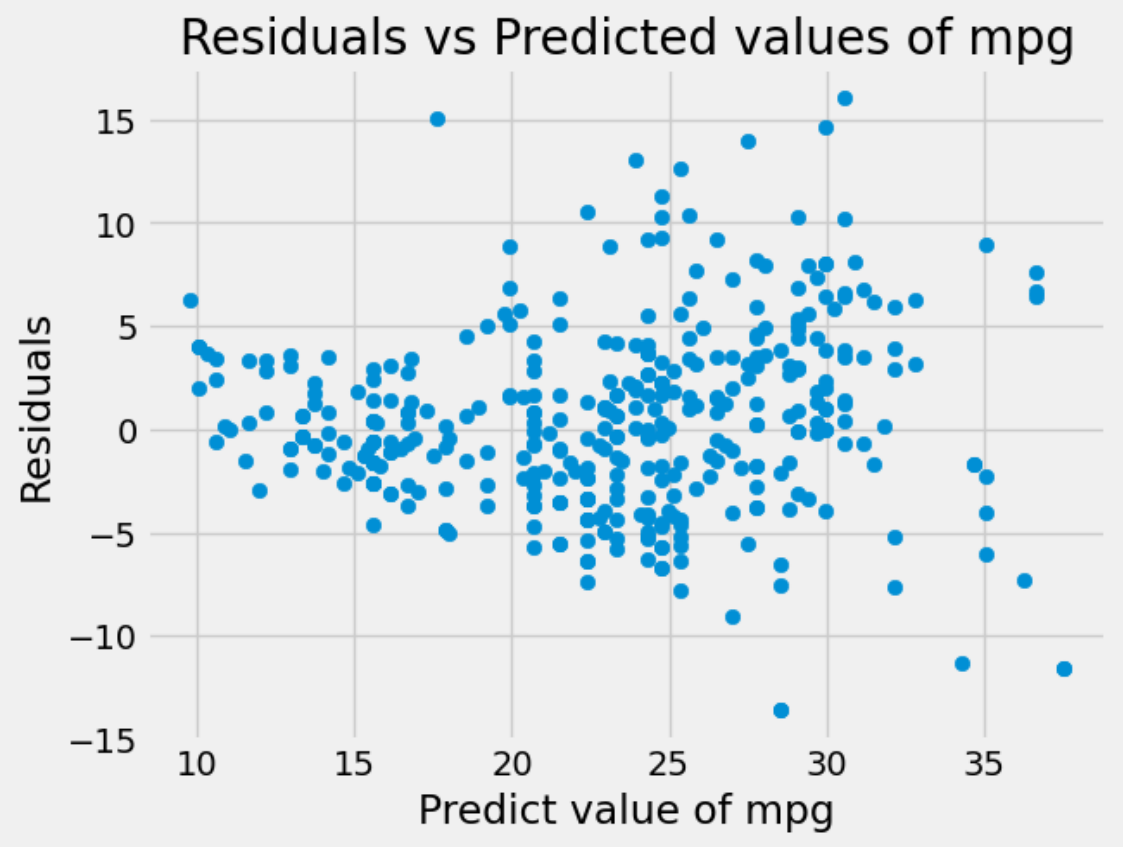


Question 4.7:

```
In [51]: # a)
         model3 = lm.LinearRegression()
         x = vehicle_data[['sqrt(hp)']]
         y = vehicle_data['log(mpg)']
         model3.fit(x, y)
         # b)
         predicted_Model3 = np.exp(model3.predict(x))

         # Your code for parts a) and b) above this line
```

```
In [52]: plt.scatter(vehicle_data['sqrt(hp)'], vehicle_data['mpg'], label = 'Orignal' , color = 'red')
         plt.plot(vehicle_data['sqrt(hp)'], predicted_Model3,
                  label=r'$mpg = e^{\theta_0 + \theta_1 \sqrt{horsepower}}$',
                  color = 'green')
         plt.xlabel('sqrt(horsepower)')
         plt.ylabel('mpg')
         plt.legend()
         plt.title('Log of mpg vs sqrt(horsepower)')
         plt.show()

         residual_Model3 = vehicle_data['mpg'] - predicted_Model3
         plt.scatter(predicted_Model3, residual_Model3)
         plt.xlabel('Predict value of mpg')
         plt.ylabel('Residuals')
         plt.title('Residuals vs Predicted values of mpg')
         plt.show()

         # Your code for part c  above this line
```


Log of mpg vs sqrt(horsepower)
Orignal
$mpg = e^{\theta_0 + \theta_1 \sqrt{horsepower}}$
mpg
sqrt(horsepower)
45
40
35
30
25
20
15
10
8
10
12
14




```
In [53]: RMSE_model3 = np.sqrt(((vehicle_data['mpg'] - predicted_Model3)**2).mean())

         print("The RMSE of this model is ", RMSE_model3)
```

```
The RMSE of this model is  4.462359658070861
```

```
In [54]: grader.check("q4_7")
```

```
Out[54]: q4_7 results: All test cases passed!
```

Question 4.8. Analyze this new model compared to the first 2 ones. Which RMSE is smaller? Does the residual plot of this new model indicate this model is a better choice than the first two? Why or why not?

This new model has an even smaller RMSE than the first 2 models. This means that it is better than the choice of the first two, as scatterplot seems more randomnize which shows that it has reduced errors than the orignal data. This shows a better choice for predicting the relationship between horsepower and mpg.