(1) For the questions 4,7,8. We changed the code to keep three significant digits after the decimal place:

Q4.

```
female = len(data[data['Gender'] == 'Female'])
male = len(data[data['Gender'] == 'Male'])
a4 = round(female / male, 3)
```

Q7.

```
a7 = round(float(data['Usage'].mean()), 3)
```

Q8.

```
a8 = round(len(data[data['Fitness'] == 5]) / len(data), 3)
```

Output:

```
# Test your class. For example run:
for question, answer in answers.items():
    print(f"Answer to Question {question}: {answer}")
```

executed in 4ms, finished 13:20:32 2024-12-10

```
Answer to Question 1: 180
Answer to Question 2: 3
Answer to Question 3: 32
Answer to Question 4: 0.731
Answer to Question 5: 16.0
Answer to Question 6: 107
Answer to Question 7: 3.456
Answer to Question 8: 0.172
Answer to Question 9: 104581
Answer to Question 10: 28
```

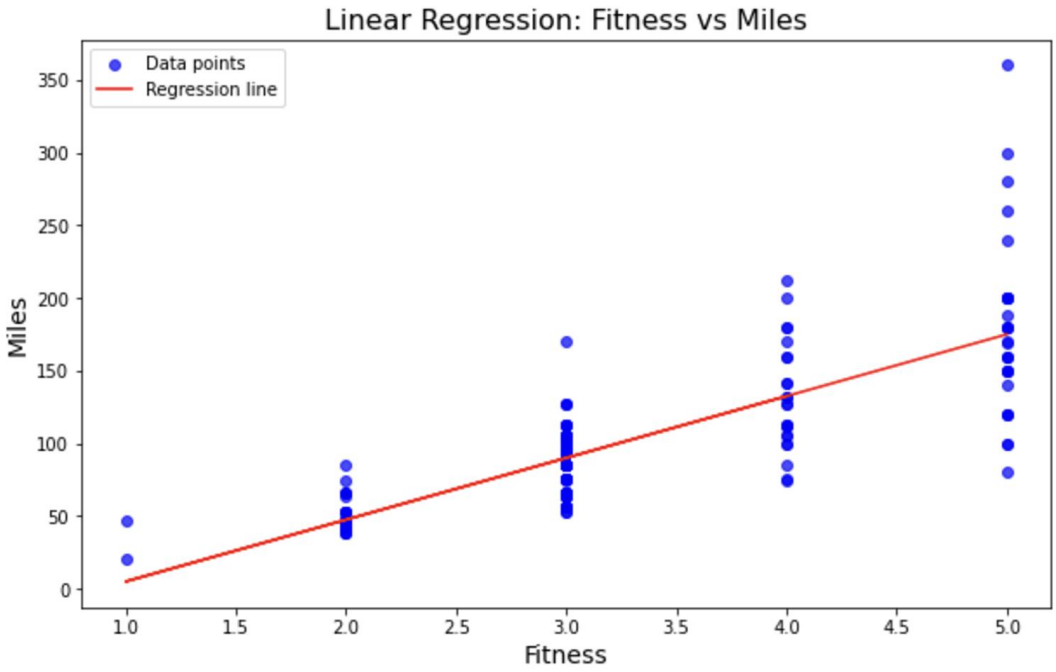
(2) For the second graph: we changed the codes to use the simple code to get the same result:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
import numpy as np
```

```
# Step 1: Read the CSV file into a pandas DataFrame
file_name = "CardioGoodFitness.csv"
data = pd.read_csv(file_name)

# Step 2: Extract relevant columns (Age and Miles)
age = data["Age"]
miles = data["Miles"]

# Step 3: Plot the linear regression diagram using seaborn
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

# Create the regression plot
sns.regplot(x=age, y=miles, data=data, scatter_kws={"color": "blue"}, line_kws={"color": "red"})

# Step 4: Add plot title and labels
plt.title("Linear Regression: Age vs Miles", fontsize=16)
plt.xlabel("Age", fontsize=14)
plt.ylabel("Miles", fontsize=14)

# Display the plot
plt.show()
```

Output:

```
Slope (m): 42.4972994869025
Intercept (b): -37.51883607885493
```

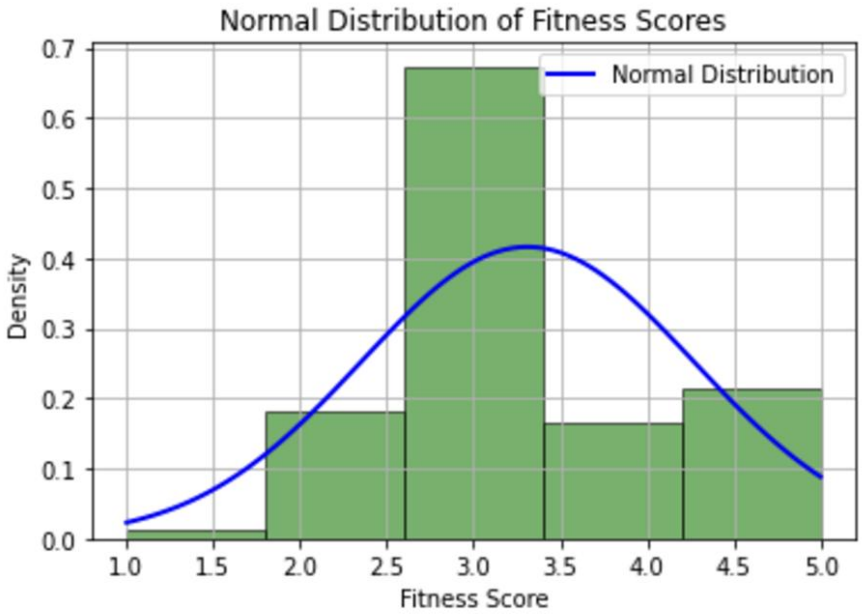


(3) For the third normal distribution graph, we added more code to verify whether the data has a normal distribution and output the results:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Assuming 'data' is your DataFrame and 'Fitness' column exists

# Get the 'Fitness' column and remove any missing values
fitness_data = data['Fitness'].dropna()

# Compute the mean and standard deviation of the 'Fitness' column
mean_fitness = fitness_data.mean()
std_fitness = fitness_data.std()

# Create an array of values from the minimum to the maximum fitness score
x = np.linspace(fitness_data.min(), fitness_data.max(), 100)

# Calculate the normal distribution values based on mean and std
normal_dist = (1 / (std_fitness * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - mean_fitness) / std_fitness) ** 2)

# Plot histogram of fitness scores
plt.hist(fitness_data, bins=5, density=True, alpha=0.6, color='g', edgecolor='black')

# Overlay the normal distribution curve
plt.plot(x, normal_dist, label='Normal Distribution', color='blue', linewidth=2)

# Customize the plot
plt.title('Normal Distribution of Fitness Scores')
plt.xlabel('Fitness Score')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.show()

# Manual skewness and kurtosis calculation
n = len(fitness_data)
mean_diff = fitness_data - mean_fitness
skewness = (n * sum(mean_diff**3)) / ((n - 1) * (n - 2) * std_fitness**3)
kurtosis = (n * (n + 1) * sum(mean_diff**4)) / ((n - 1) * (n - 2) * (n - 3) * std_fitness**4) - (3 * (n - 1)**2) / ((n - 2) * (n - 3))
```

```
# Print results
print(f"Mean of Fitness: {mean_fitness}")
print(f"Standard Deviation of Fitness: {std_fitness}")
print(f"Skewness: {skewness:.2f}")
print(f"Kurtosis: {kurtosis:.2f}")

# Interpretation of skewness and kurtosis
if abs(skewness) < 0.5:
    print("The data is approximately symmetric.")
elif skewness > 0.5:
    print("The data is positively skewed (longer tail on the right).")
else:
    print("The data is negatively skewed (longer tail on the left).")

if abs(kurtosis) < 1:
    print("The data has a kurtosis close to that of a normal distribution.")
elif kurtosis > 1:
    print("The data has heavier tails (leptokurtic).")
else:
    print("The data has lighter tails (platykurtic).")
```

Output:



Mean of Fitness: 3.311111111111111
Standard Deviation of Fitness: 0.958868565619312
Skewness: 0.45
Kurtosis: -0.37
The data is approximately symmetric.
The data has a kurtosis close to that of a normal distribution.