Iris Dataset

2. Using the iris dataset...

- a. Make a histogram of the variable Sepal.Width.
- b. Based on the histogram from #1a, which would you expect to be higher, the mean or the median? Why?
- c. Confirm your answer to #1b by actually finding these values.
- d. Only 27% of the flowers have a Sepal.Width higher than _____ cm.
- e. Make scatterplots of each pair of the numerical variables in iris (There should be 6 pairs/plots).
- f. Based on #1e, which two variables appear to have the strongest relationship? And which two appear to have the weakest relationship?

```
from sklearn import datasets
iris = datasets.load_iris(as_frame=True)
```

✓ [4] 819ms

```
# The Bunch object still holds the metadata
X_df = iris.data
y_df = iris.target

# Combine features and target into a single DataFrame for easier viewing
full_df = X_df.copy()
full_df['species'] = y_df

full_df.head()
```

✓ [5] 27ms

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

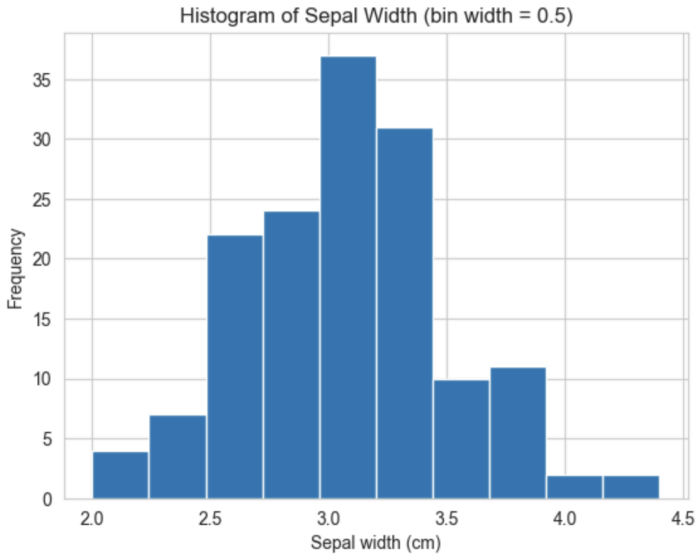
a. Make a histogram of the variable Sepal.Width.

```
import matplotlib.pyplot as plt
# histogram
full_df['sepal width (cm)'].hist()

# Labels + title
plt.xlabel('Sepal width (cm)')
plt.ylabel('Frequency')
plt.title('Histogram of Sepal Width (bin width = 0.5)')
```

✓ [9] 392ms

```
Text(0.5, 1.0, 'Histogram of Sepal Width (bin width = 0.5)')
```



b. Based on the histogram from #1a, which would you expect to be higher, the mean or the median? Why?

The histogram looks pretty close to a normal curve, which suggests the mean and median are about the same.

c. Confirm your answer to #1b by actually finding these values.

As expected the mean and median are approximately the same as seen in the table below (mean = 3.06 and the median = 3.0)

```
full_df.describe()
```
[19]

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

d. Only 27% of the flowers have a Sepal.Width higher than _____ cm.

We are looking for the 73rd percentile of Sepal.Width (100% – 27% = 73%). The value is 3.3 cm. Alternatively, we could use the 75th percentile from the table above as an approximation.

```
threshold = full_df['sepal width (cm)'].quantile(0.73)
threshold
```
[21]

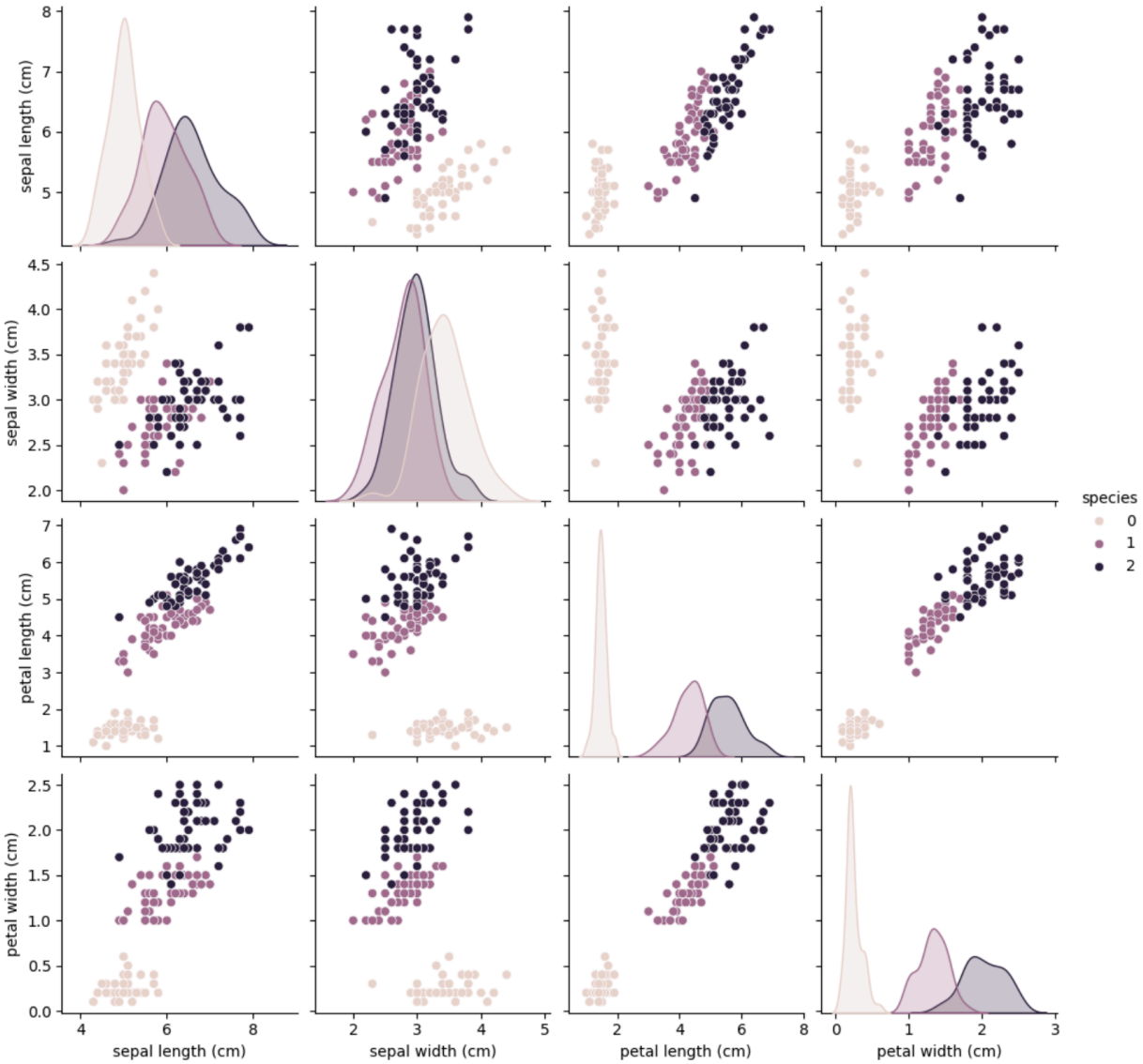
```
np.float64(3.3)
```

e. Make scatterplots of each pair of the numerical variables in iris (There should be 6 pairs/plots)

- Sepal.Length vs Sepal.Width
- Sepal.Length vs Petal.Length
- Sepal.Length vs Petal.Width
- Sepal.Width vs Petal.Length
- Sepal.Width vs Petal.Width
- Petal.Length vs Petal.Width

```
import seaborn as sns
sns.pairplot(data=full_df, hue="species")
```
[32]

```
<seaborn.axisgrid.PairGrid at 0x11ac5fb10>
```


sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
species
0
1
2

f. Based on #1e, which two variables appear to have the strongest relationship? And which two appear to have the weakest relationship?

Petal length and petal width appear to have the strongest relationship, while sepal length and sepal width show the weakest relationship.

```
# Compute the correlation matrix
corr = full_df.corr()
corr
```
✓ [12] 18ms

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
sepal length (cm)	1.000000	-0.117570	0.871754	0.817941	0.782561
sepal width (cm)	-0.117570	1.000000	-0.428440	-0.366126	-0.426658
petal length (cm)	0.871754	-0.428440	1.000000	0.962865	0.949035
petal width (cm)	0.817941	-0.366126	0.962865	1.000000	0.956547
species	0.782561	-0.426658	0.949035	0.956547	1.000000

Plantgrowth Dataset

- a. Make a histogram of the variable weight with breakpoints (bin edges) at every 0.3 units, starting at 3.3.
- b. Make boxplots of weight separated by group in a single graph.
- c. Based on the boxplots in #2b, approximately what percentage of the "trt1" weights are below the minimum "trt2" weight?
- d. Find the exact percentage of the "trt1" weights that are below the minimum "trt2" weight.
- e. Only including plants with a weight above 5.5, make a barplot of the variable group. Make the barplot colorful using some color palette (in R, try running ?heat.colors and/or check out https://www.r-bloggers.com/palettes-in-r/).

```
import pandas as pd
data = { "weight": [4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14, 4.81, 4.17, 4.41, 3.59, 5.87,
  3.83, 6.03, 4.89, 4.32, 4.69, 6.31, 5.12, 5.54, 5.50, 5.37, 5.29, 4.92, 6.15, 5.80, 5.26], "group":
 ["ctrl"] * 10 + ["trt1"] * 10 + ["trt2"] * 10}
PlantGrowth = pd.DataFrame(data)
PlantGrowth.head()
```
✓ [17] 10ms

	weight	group
0	4.17	ctrl
1	5.58	ctrl
2	5.18	ctrl
3	6.11	ctrl
4	4.50	ctrl

a. Make a histogram of the variable weight with breakpoints (bin edges) at every 0.3 units, starting at 3.3.

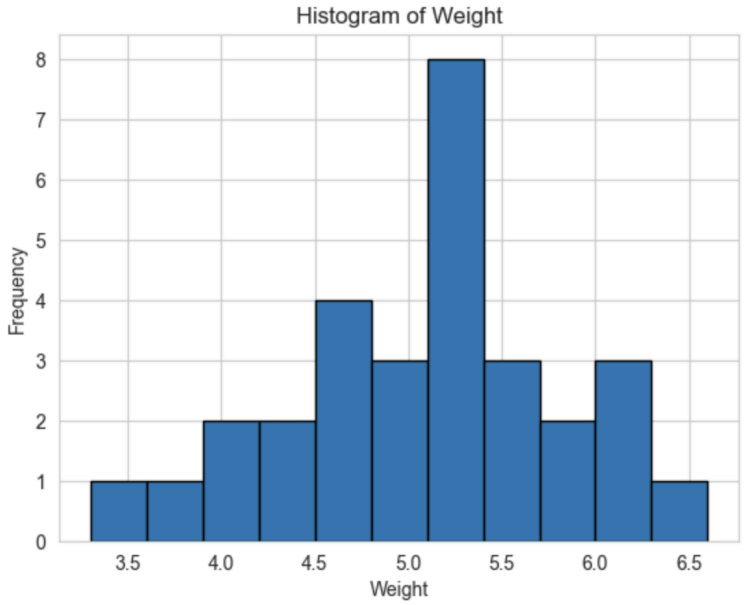
```python
import numpy as np

# Define bin edges from 3.3 to the max weight, step 0.3
bins = np.arange(3.3, max(PlantGrowth['weight']) + 0.3, 0.3)

# Plot the histogram
plt.hist(PlantGrowth['weight'], bins=bins, edgecolor='black')
plt.xlabel('Weight')
plt.ylabel('Frequency')
plt.title('Histogram of Weight')
plt.show()

```

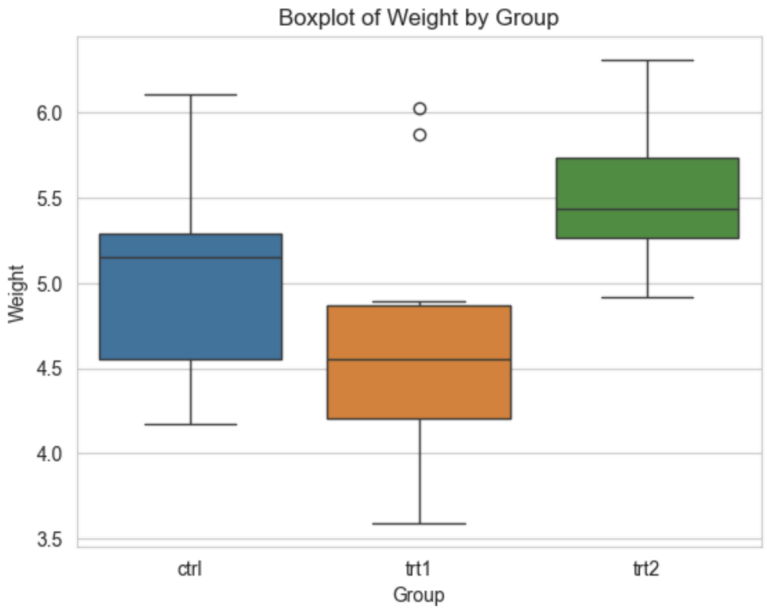
✓ [20] 160ms



b. Make boxplots of weight separated by group in a single graph.

```
import seaborn as sns
# Boxplot
sns.boxplot(x='group', y='weight', hue='group', data=PlantGrowth, palette='tab10')
plt.xlabel('Group')
plt.ylabel('Weight')
plt.title('Boxplot of Weight by Group')
plt.show()
```

✓ [28] 141ms



c. Based on the boxplots in #2b, approximately what percentage of the "trt1" weights are below the minimum "trt2" weight?

From the boxplots, it seems that almost 100% of the trt1 weights fall below the minimum trt2 weight.

d. Find the exact percentage of the "trt1" weights that are below the minimum "trt2" weight.

Exactly 80% of trt1 weights are below the minimum trt2 weight.

```
trt2 = [6.31, 5.12, 5.54, 5.50, 5.37, 5.29, 4.92, 6.15, 5.80, 5.26]
min_trt2 = min(trt2)  # 4.92
print(min_trt2)

trt1 = [4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69]
below_min = [w for w in trt1 if w < min_trt2]
len(below_min)  # 8
```

✓ [30] < 10 ms

```
4.92
8
```

e. Only including plants with a weight above 5.5, make a barplot of the variable group. Make the barplot colorful using some color palette.

```python
# Filter for plants with weight > 5.5
df_filtered = PlantGrowth[PlantGrowth['weight'] > 5.5]

# Barplot of group counts with colors
sns.countplot(x='group', data=df_filtered, hue=df_filtered['group'], palette='Set2')
plt.xlabel('Group')
plt.ylabel('Count')
plt.title('Number of Plants with Weight > 5.5 by Group')
plt.show()
```

✓ [32] 222ms