In [64]:

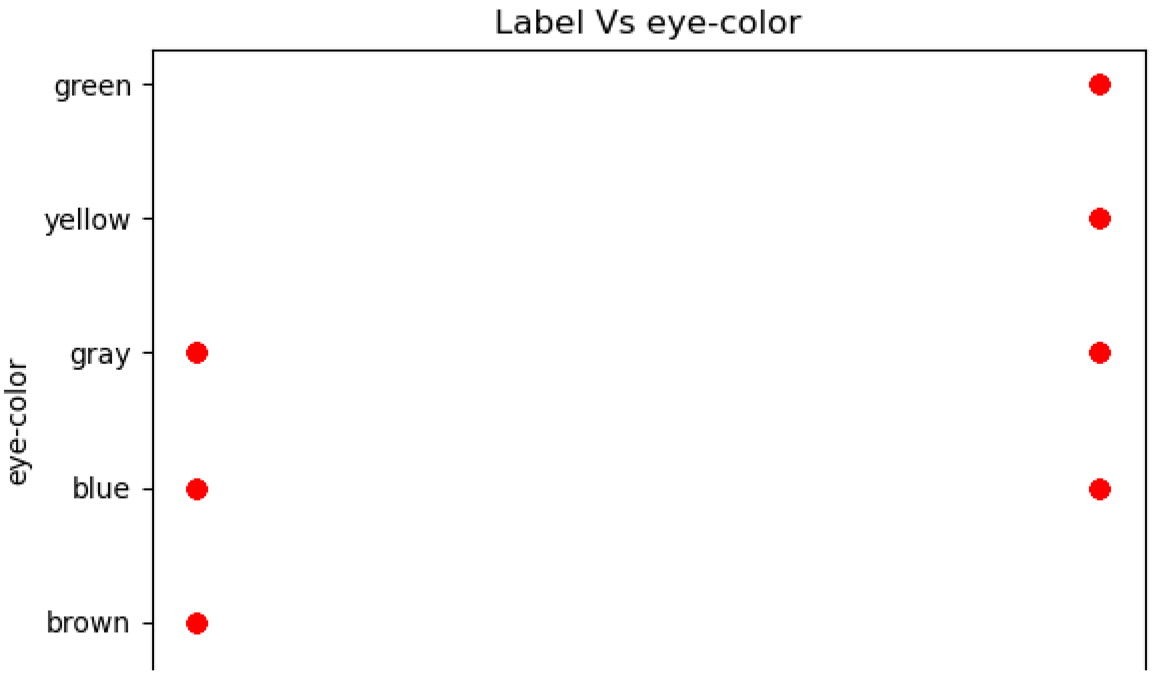
```
#Note: For all tasks you need to load data from given excel file.

#Basic data cleaning and neccessary imports
!pip install ydata-profiling
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
data = pd.read_csv('Cat_human_New.csv')

# Remove duplicate columns
data = data.loc[:, ~data.columns.duplicated()]

# Remove duplicate rows
data = data.drop_duplicates()

# Remove blank columns
data = data.dropna(axis=1, how='all')

# Remove blank rows
data = data.dropna(axis=0, how='all')

# Display the cleaned data
data
```

```
Collecting ydata-profiling
  Downloading ydata_profiling-4.2.0-py2.py3-none-any.whl (352 kB)
     ------------------------------------ 352.3/352.3 kB 245.9 kB/s eta 0:00:00
Collecting phik<0.13,>=0.11.1
  Downloading phik-0.12.3-cp310-cp310-win_amd64.whl (663 kB)
     ------------------------------------ 663.4/663.4 kB 182.5 kB/s eta 0:00:00
Requirement already satisfied: numpy<1.24,>=1.16.0 in c:\users\barcha\anaconda3\lib\site-packages (from y
data-profiling) (1.23.5)
Requirement already satisfied: tqdm<5,>=4.48.2 in c:\users\barcha\anaconda3\lib\site-packages (from ydata
-profiling) (4.64.1)
Collecting imagehash==4.3.1
  Downloading ImageHash-4.3.1-py2.py3-none-any.whl (296 kB)
     ------------------------------------ 296.5/296.5 kB 145.4 kB/s eta 0:00:00
Requirement already satisfied: statsmodels<1,>=0.13.2 in c:\users\barcha\anaconda3\lib\site-packages (fro
m ydata-profiling) (0.13.5)
Collecting typeguard<3,>=2.13.2
  Downloading typeguard-2.13.3-py3-none-any.whl (17 kB)
Requirement already satisfied: matplotlib<4,>=3.2 in c:\users\barcha\anaconda3\lib\site-packages (from yd
ata-profiling) (3.7.0)
Collecting visions[type_image_path]==0.7.5
```

# Task1

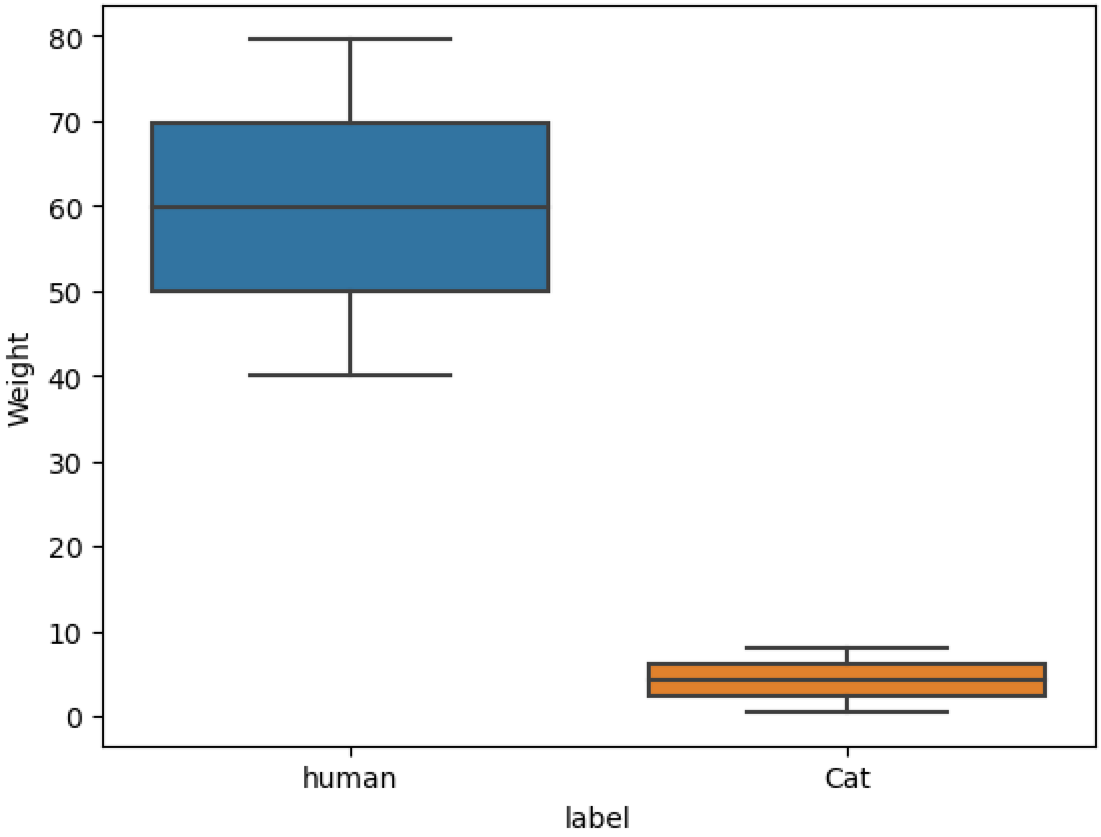
In [24]:

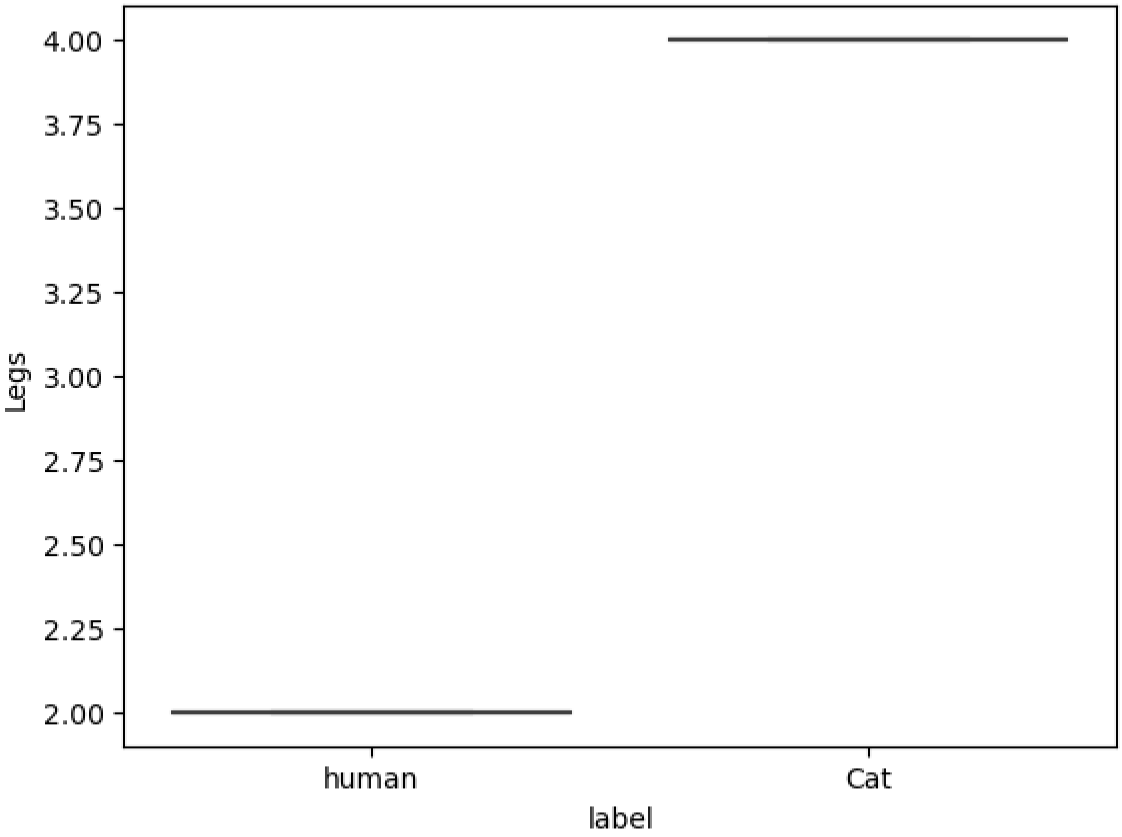
```
#Read given cat_dog.csv file into dataframe. Display different scatterplots between
#all features.
#⍰ Each column visualization

data['Weight'] = pd.to_numeric(data['Weight'], errors='coerce')



plt.title('Label Vs Weights')
plt.xlabel('Label')
plt.ylabel('Weights')
plt.scatter(data['label'],data['Weight'],color = 'red')
plt.show()

plt.title('Label Vs color')
plt.xlabel('Label')
plt.ylabel('color')
plt.scatter(data['label'],data['Color'],color = 'red')
plt.show()

plt.title('Label Vs eye-color')
plt.xlabel('Label')
plt.ylabel('eye-color')
plt.scatter(data['label'],data['Eye_color'],color = 'red')
plt.show()

plt.title('Label Vs eye-color')
plt.xlabel('Label')
plt.ylabel('eye-color')
plt.scatter(data['label'],data['Eye_color'],color = 'red')
plt.show()

plt.title('Label Vs Heights')
plt.xlabel('Label')
plt.ylabel('Heights')
plt.scatter(data['label'],data['Height'],color = 'red')
plt.show()

plt.title('Label Vs Legs')
plt.xlabel('Label')
plt.ylabel('Legs')
plt.scatter(data['label'],data['Legs'],color = 'red')
plt.show()
plt.title('Label Vs Moustache')
plt.xlabel('Label')
plt.ylabel('Moustache')
plt.scatter(data['label'],data['Moustache'],color = 'red')
plt.show()

plt.title('Label Vs Tail')
plt.xlabel('Label')
plt.ylabel('Tail')
plt.scatter(data['label'],data['Tail'],color = 'red')
plt.show()
```

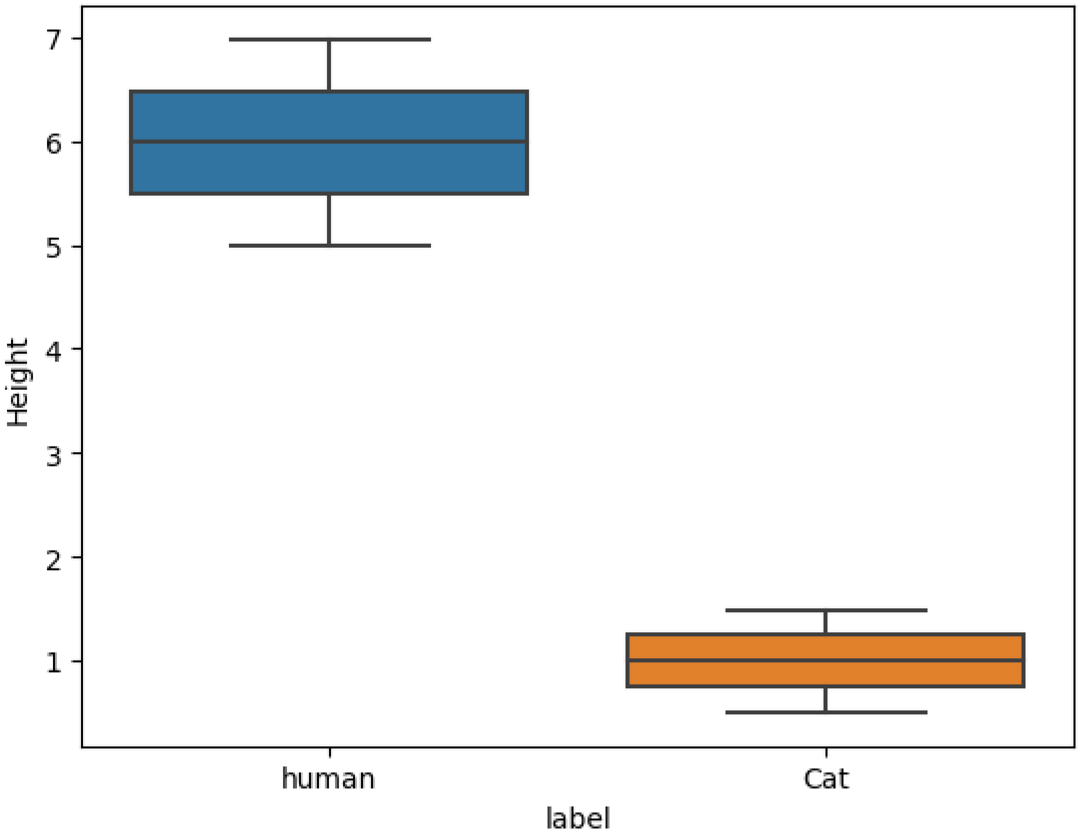


In [106]:

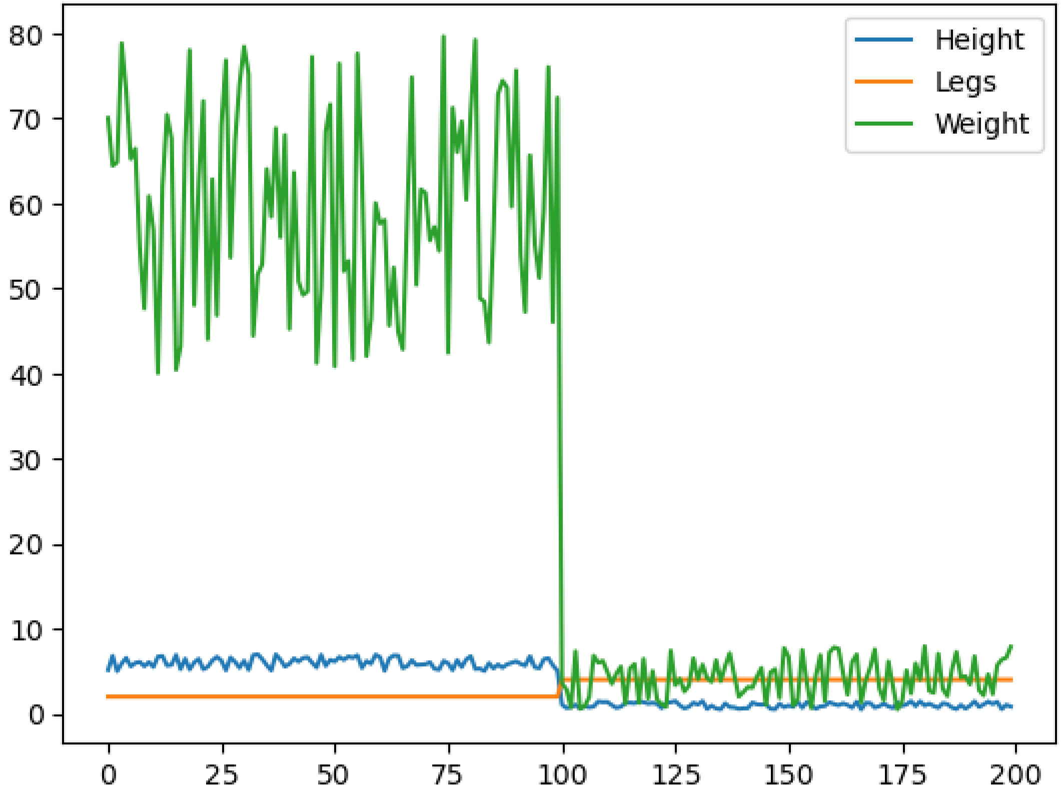
```
#□ Draw box plot


sb.boxplot(data=data, x="label", y="Weight")
plt.show()
sb.boxplot(data=data, x="label", y="Legs")
plt.show()
sb.boxplot(data=data, x="label", y="Height")
plt.show()
```




4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00
Legs
human
Cat
label

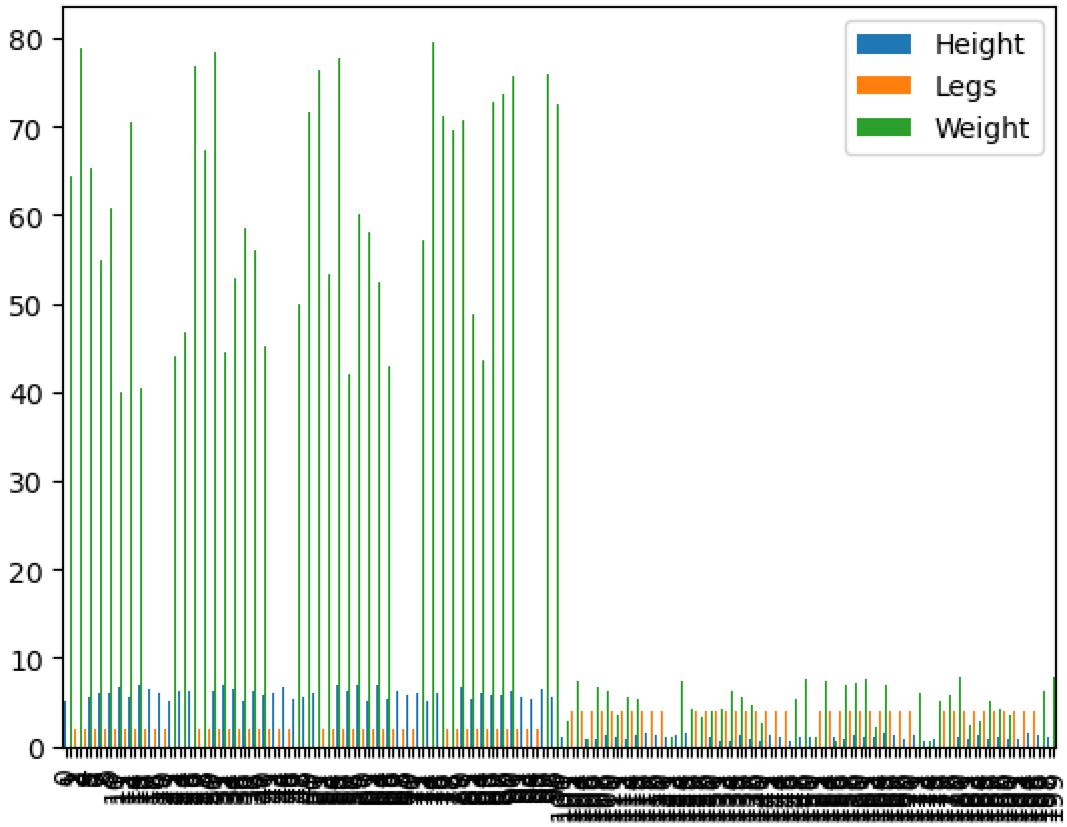

7
6
5
4
3
2
1
Height
human
Cat
label

```
In [26]: #⍰ Draw Line Graph
         data.plot(kind='line')

Out[26]: <Axes: >
```



In [108]:
```
#□ Draw Bar Graph

  # Increase the figure size as desired
dat = pd.DataFrame(data)
dat.plot(kind='bar')
```

Out[108]:
```
<Axes: >
```

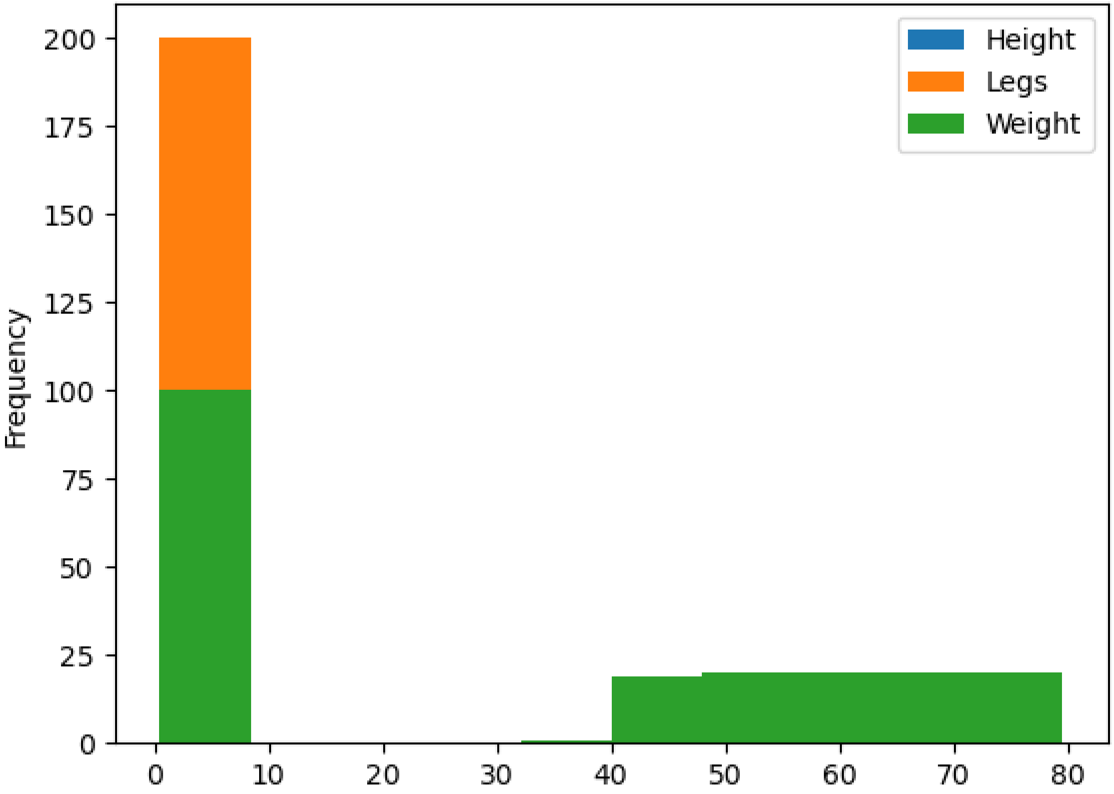


In [104]:

```
#⍰ Draw Histogram
dat = pd.DataFrame(data)
dat.plot(kind='hist')
```

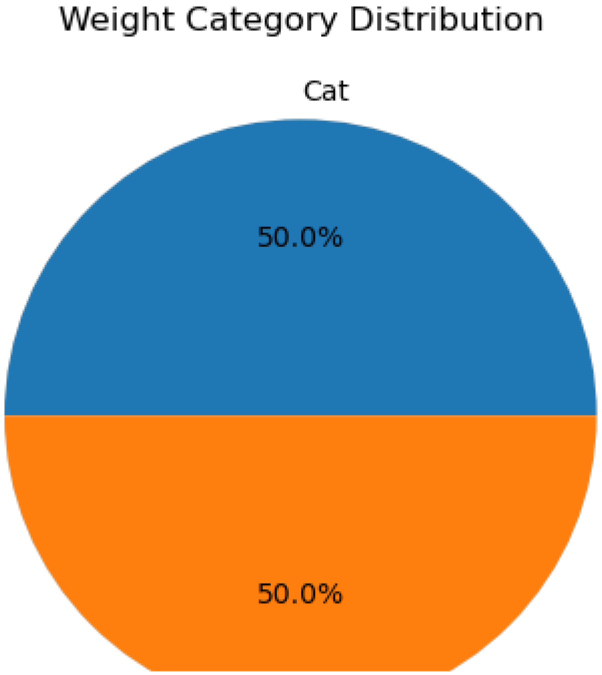
Out[104]:

```
<Axes: ylabel='Frequency'>
```



In [134]:

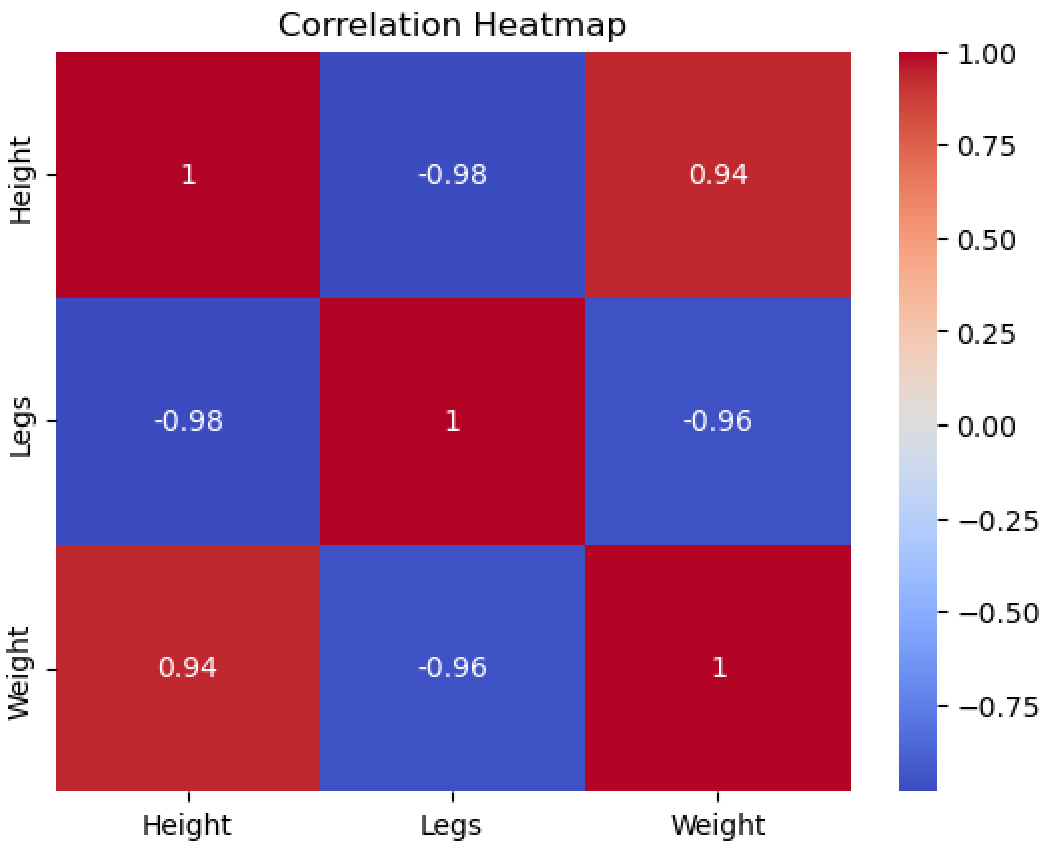
```
eye = data['label'].value_counts()
plt.title('label')
plt.pie(eye.values, labels=eye.index, autopct='%1.1f%%')
plt.show()


eye = data['Legs'].value_counts()
plt.title('Legs')
plt.pie(eye.values, labels=eye.index, autopct='%1.1f%%')
plt.show()

# Categorize data based on weight condition
cat_count = data[data['Weight'] < 10].shape[0]
human_count = data[data['Weight'] >= 10].shape[0]

# Create labels for the pie chart
labels = ['Cat', 'Human']

# Create data for the pie chart
sizes = [cat_count, human_count]

# Plot the pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%')

# Set the title of the pie chart
plt.title('Weight Category Distribution')

# Display the chart
plt.show()


eye = data['Moustache'].value_counts()
plt.title('Moustache')
plt.pie(eye.values, labels=eye.index, autopct='%1.1f%%')
plt.show()

eye = data['Tail'].value_counts()
plt.title('Tail')
plt.pie(eye.values, labels=eye.index, autopct='%1.1f%%')
plt.show()

eye = data['Eye_color'].value_counts()
plt.title('Eye_color')
plt.pie(eye.values, labels=eye.index, autopct='%1.1f%%')
plt.show()

eye = data['Color'].value_counts()
plt.title('Color')
plt.pie(eye.values, labels=eye.index, autopct='%1.1f%%')
plt.show()
```



```
In [ ]:
```

In [80]:

```
#Draw Heat map
import seaborn as sb

# Select only the numerical columns for the heatmap
numerical_data = data.select_dtypes(include='number')

# Create the heatmap
sb.heatmap(numerical_data.corr(), annot=True, cmap='coolwarm')

# Set the title of the heatmap
plt.title('Correlation Heatmap')

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



In [77]:

```
#⍰ Write a complete report on this dataset
data = pd.read_csv('Cat_human_New.csv')

report = data.describe()
print('before cleaning')
print(data)

# Remove duplicate columns
data = data.loc[:, ~data.columns.duplicated()]

# Remove duplicate rows
data = data.drop_duplicates()

# Remove blank columns
data = data.dropna(axis=1, how='all')

# Remove blank rows
data = data.dropna(axis=0, how='all')

# Display the cleaned data

report = data.describe()
print('After cleaning')
print(data)
```

```
before cleaning
           Color Eye_color  Height  Legs Moustache Tail     Weight  label
0          black     black    5.14     2        No   No  70.000000  human
1     dark_brown     brown    6.80     2        No   No  64.400000  human
2    light_brown     brown    5.00     2       Yes   No  64.800000  human
3    light_brown      blue    5.90     2        No   No  78.800000  human
4    light_brown      blue    6.56     2        No   No  73.200000  human
..           ...       ...     ...   ...       ...  ...        ...    ...
195        brown      gray    1.14     4       Yes  Yes   2.304511    Cat
196        white    yellow    1.39     4       Yes  Yes   5.687970    Cat
197        white     black    0.53     4       Yes  Yes   6.364662    Cat
198        brown     green    1.03     4       Yes  Yes   6.590226    Cat
199  brown_white      blue    0.83     4       Yes  Yes   7.868421    Cat

[200 rows x 8 columns]
After cleaning
           Color Eye_color  Height  Legs Moustache Tail     Weight  label
0          black     black    5.14     2        No   No  70.000000  human
1     dark_brown     brown    6.80     2        No   No  64.400000  human
2    light_brown     brown    5.00     2       Yes   No  64.800000  human
3    light_brown      blue    5.90     2        No   No  78.800000  human
4    light_brown      blue    6.56     2        No   No  73.200000  human
..           ...       ...     ...   ...       ...  ...        ...    ...
195        brown      gray    1.14     4       Yes  Yes   2.304511    Cat
196        white    yellow    1.39     4       Yes  Yes   5.687970    Cat
197        white     black    0.53     4       Yes  Yes   6.364662    Cat
198        brown     green    1.03     4       Yes  Yes   6.590226    Cat
199  brown_white      blue    0.83     4       Yes  Yes   7.868421    Cat

[200 rows x 8 columns]
```

In [ ]:

# Task2

In [90]:

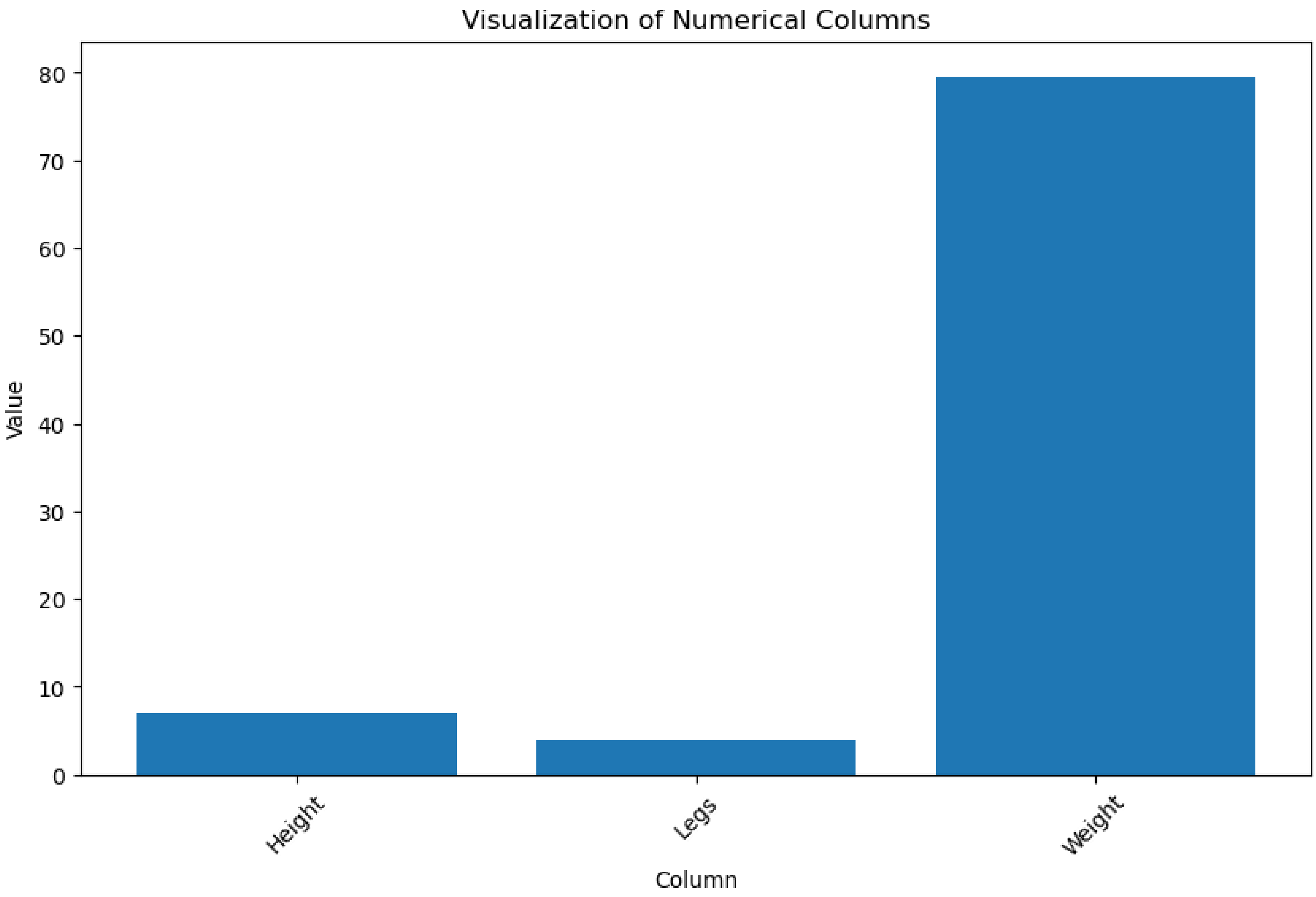
```
#Task2:6
#Make pandas profiling
#Convert each column into row and perform visualization
#perform visualization on numerical data and categorical data separately
import pandas_profiling as pp

# Generate a pandas profiling report for the DataFrame
profile = pp.ProfileReport(data)
profile.to_file("reportCatHuman.html")
```

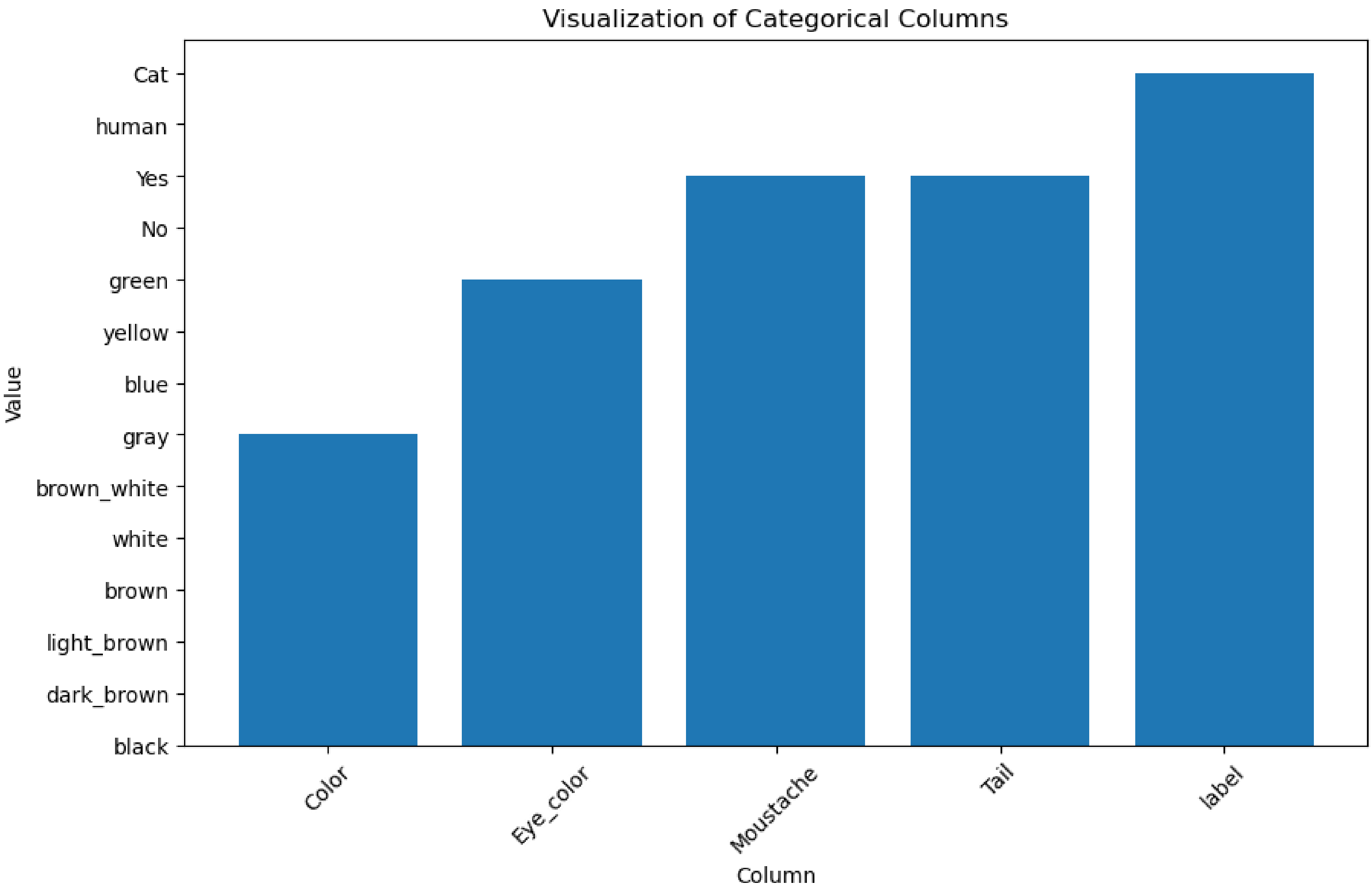
```
Generate report structure: 100%                                   1/1 [00:09<00:00, 9.55s/it]

Render HTML: 100%                                           1/1 [00:01<00:00, 1.46s/it]

Export report to file: 100%                                    1/1 [00:00<00:00, 7.62it/s]
```

In [ ]:

In [71]:

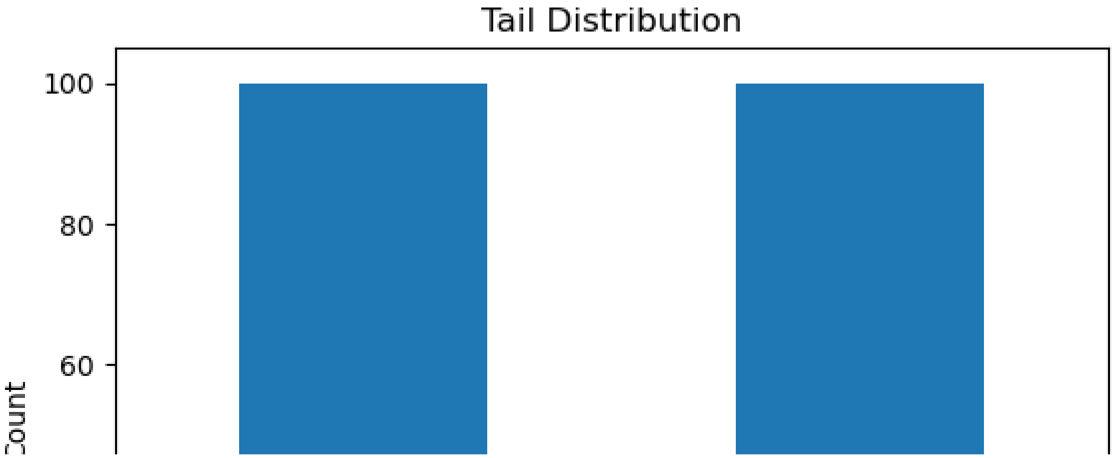
```
import pandas as pd
import matplotlib.pyplot as plt

# Convert each column into rows for numerical data
numerical_data = data.select_dtypes(include='number')
melted_numerical_data = numerical_data.melt(var_name='Column', value_name='Value')

# Perform visualization on melted numerical data
plt.figure(figsize=(10, 6))
plt.bar(melted_numerical_data['Column'], melted_numerical_data['Value'])
plt.xlabel('Column')
plt.ylabel('Value')
plt.title('Visualization of Numerical Columns')
plt.xticks(rotation=45)
plt.show()

# Convert each column into rows for categorical data
categorical_data = data.select_dtypes(include='object')
melted_categorical_data = categorical_data.melt(var_name='Column', value_name='Value')

# Perform visualization on melted categorical data
plt.figure(figsize=(10, 6))
plt.bar(melted_categorical_data['Column'], melted_categorical_data['Value'])
plt.xlabel('Column')
plt.ylabel('Value')
plt.title('Visualization of Categorical Columns')
plt.xticks(rotation=45)
plt.show()
```


Visualization of Numerical Columns
Value
0
10
20
30
40
50
60
70
80
Height
Legs
Weight
Column


Visualization of Categorical Columns
Value
Cat
human
Yes
No
green
yellow
blue
gray
brown_white
white
brown
light_brown
dark_brown
black
Color
Eye_color
Moustache
Tail
label
Column

In [93]:
```
import matplotlib.pyplot as plt

# Convert each column into a row and perform visualization
for column in data.columns:
    plt.figure()
    data[column].value_counts().plot(kind='bar')
    plt.title(f"{column} Distribution")
    plt.xlabel(column)
    plt.ylabel("Count")
    plt.show()
```

0
Yes
No
Moustache



In [ ]:

In [ ]:

In [ ]: