Veermata Jijabai Technological Institute

(An Autonomous Institute of Government of Maharashtra)

Department: Electronics and Telecommunication Engineering

(Data Science and Analysis Lab- R4ET3104P)

Experiment No.2

Aim: Concept of Data Wrangling.

Name of Students: 1. Nandini Pramod Junghare (211091012)

2. Astha Shankar Shinde (211091044)

Year & Semester: Third year sixth semester

Branch: Electronics and Telecommunication

EXPERIMENT NO. 2

Aim: To understand various concepts about Data Wrangling in python.

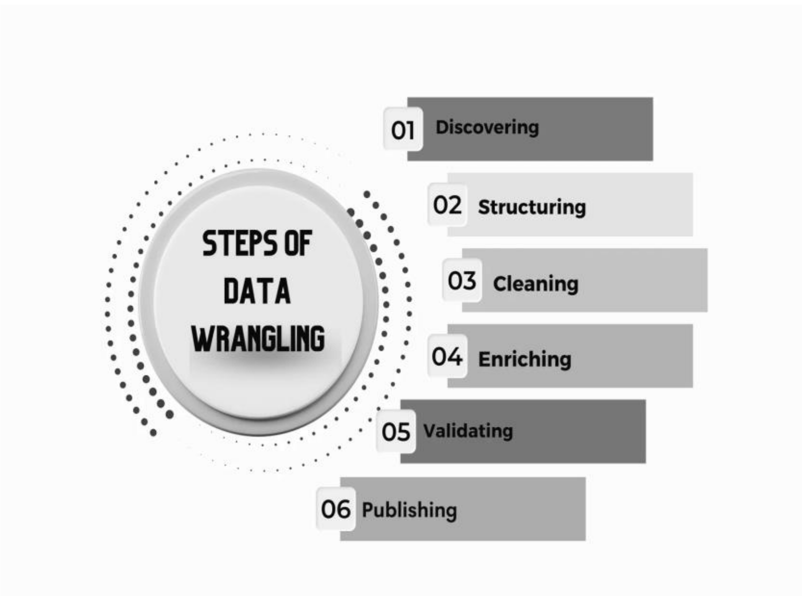
Objective: The objective of this experiment is to demonstrate the process of data wrangling on a real-world dataset and prepare it for further analysis.

Software used: Jupyter Notebook.

Theory: Data wrangling is a crucial step in the data science workflow, involving the cleaning, transforming, and preparing of raw data for analysis. In this report, we document our data wrangling experiment conducted using Python.

Data Wrangling in Python:
Data Wrangling is a crucial topic for Data Science and Data Analysis. Panda Framework of Python is used for Data Wrangling. Pandas is an open-source library in Python specifically developed for Data Analysis and Data Science. It is used for processes like data sorting or filtration, Data grouping, etc.
Data wrangling in Python deals with the below functionalities:

- ➔ Data exploration: In this process, the data is studied, analyzed, and understood by visualizing representation of data.
- ➔ Dealing with missing values: Most of the datasets having a vast amount of data contain missing values of *NAN*, they are needed to be taken care of by replacing them with mean, mode, the most frequent value of the column, or simply by dropping the row having a *NAN* value.
- ➔ Reshaping data: In this process, data is manipulated according to the requirements, where new data can be added or pre-existing data can be modified.
- ➔ Filtering data: Sometimes datasets are comprised of unwanted rows or columns which are required to be removed or filtered.
- ➔ Other: After dealing with the raw dataset with the above functionality we get an efficient dataset as per our requirements and then it can be used for a required purpose like data analyzing, machine learning, data visualization, model training etc.



Methodology:

1. Data Acquisition: We downloaded the dataset and loaded it into our Python environment using pandas.

2. Data Exploration: We conducted an initial exploration of the dataset to understand its structure, features, and any missing values.

3. Data Cleaning:

- Handling Missing Values: We identified missing values in the dataset and applied techniques such as imputation or removal based on the context of the data.

- Handling Outliers: We detected outliers and decided whether to remove them or treat them based on domain knowledge.

- Handling Inconsistencies: We checked for inconsistencies in categorical variables and corrected them if necessary.

4. Data Transformation:

- Feature Engineering: We created new features from existing ones

- Data Encoding: We encoded categorical variables into numerical format using techniques like one-hot encoding or label encoding.

- Data Scaling: We scaled numerical features to ensure they have a similar range and distribution.

5. Data Analysis: We conducted exploratory data analysis (EDA) to gain insights into the relationships between variables and their impact on the target variable (survival status).

6. Data Visualization: We visualized the cleaned and transformed data using matplotlib and seaborn libraries to present key findings and patterns in the data.

In [4]:
```python
# DSA_Experiment_2
# Nandini Parmod Junghare (211091012)
# Astha Shankar Shinde (211091044)

#Data Wrangling
import pandas as pd #provides high performance, easy-to-use data structures and data ana
import numpy as np   #working with arrays
import seaborn as sns  #used to simplify graphing tasks


cols = ['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration', 'num-of-doo
```

In [5]:
```python
#Read the .csv file and store it as a pandas Data Frame
data = pd.read_csv('RAW DATA.txt',names=cols)
```

In [6]:
```python
print(data.shape)
#viewing data
data.head()
```

```
(205, 26)
```

Out[6]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	sy
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [7]:
```python
#identify missing values and REPLACE
data = data.replace("?",np.NAN)
data.head()
```

Out[7]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	sy
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [8]:
```python
#checks if there's at least one missing or null value anywhere in the data.
#If there is, it returns True; if there isn't, it returns False.
data.isnull().any().any()
```

Out[8]:
```
True
```

In [9]:
```python
#count missing values in each column
data.isnull().sum()
```

Out[9]:
```
symboling             0
normalized-losses    41
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           0
engine-type           0
num-of-cylinders      0
engine-size           0
fuel-system           0
bore                  4
stroke                4
compression-ratio     0
horsepower            2
peak-rpm              2
city-mpg              0
highway-mpg           0
price                 4
dtype: int64
```

In [35]:
```python
#objects se float kra
#calculates the average of the numbers in the 'normalized-losses' column of the dataset.
#The result is stored in a variable called avg_norm_loss.

avg_norm_loss = data['normalized-losses'].astype("float").mean()
avg_norm_loss
data["normalized-losses"].replace(np.NaN,avg_norm_loss,inplace=True)
data.head()
```

Out[35]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	sy
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

Loading [MathJax]/extensions/Safe.js

In [26]:
```
#used to count the occurrences of different values
#in the 'num-of-doors' column of the dataset.
data['num-of-doors'].value_counts()
```

Out[26]:
```
four    115
two      86
Name: num-of-doors, dtype: int64
```

In [12]:
```
#calculate mean value of bore and replcae
#This code replaces any missing values (NaN) in the "num-of-doors" column of the dataset
#The parameter inplace=True means that the changes are made directly to the dataset with
data["num-of-doors"].replace(np.nan,"four",inplace=True)
data.head()
```

Out[12]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	sy
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [13]:
```
avg_bore=data['bore'].astype("float").mean()
avg_bore
```

Out[13]:
```
3.3297512437810957
```

In [14]:
```
data["bore"].replace(np.NaN,avg_bore,inplace=True)
```

In [15]:
```
data['bore']
```

Out[15]:
```
0       3.47
1       3.47
2       2.68
3       3.19
4       3.19
        ...
200     3.78
201     3.78
202     3.58
203     3.01
204     3.78
Name: bore, Length: 205, dtype: object
```

In [16]:
```
avg_stroke=data["stroke"].astype("float").mean(axis=0)
print("Average of strokes: ",avg_stroke)

#replce NaN by mean value in 'Stroke' column
data["stroke"].replace(np.nan,avg_stroke, inplace= True)
```

```
Average of strokes:  3.2554228855721337
```

In [17]:
```
data['num-of-doors'].value_counts()
```

Loading [MathJax]/extensions/Safe.js

Out[17]:
```
four     116
two       89
Name: num-of-doors, dtype: int64
```

In [18]:
```
#This code finds the value that occurs most frequently in the "num-of-doors" column of t
#It returns the value that has the highest count.
data['num-of-doors'].value_counts().idxmax()
```

Out[18]:
```
'four'
```

In [19]:
```
data["num-of-doors"].replace(np.nan,"four", inplace=True)
data.head()
```

Out[19]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	sy
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [20]:
```
before_rows = data.shape[0]
data.dropna(subset=["price"], axis=0,inplace= True)
after_rows = data.shape[0]
```

In [33]:
```
avg_horse=data["horsepower"].astype("float").mean()
print("Average of horsepower: ",avg_horse)
data["horsepower"].replace(np.nan,avg_horse, inplace=True)
data.head()
```

```
Average of horsepower:  103.39698492462311
```

Out[33]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	sy
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [32]:
```
avg_peakrpm=data["peak-rpm"].astype("float").mean()
print("Average of peakrpm: ",avg_peakrpm)
data["peak-rpm"].replace(np.nan,avg_peakrpm, inplace=True)
data.head()
```


```
akrpm:  5117.587939698492
```

Out[32]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	sy
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [30]:

```
avg_price=data["price"].astype("float").mean()
print("Average of price: ",avg_price)
```

```
Average of price:  13207.129353233831
```

In [31]:

```
data.isnull().sum()
```

Out[31]:

```
symboling            0
normalized-losses    0
make                 0
fuel-type            0
aspiration           0
num-of-doors         0
body-style           0
drive-wheels         0
engine-location      0
wheel-base           0
length               0
width                0
height               0
curb-weight          0
engine-type          0
num-of-cylinders     0
engine-size          0
fuel-system          0
bore                 0
stroke               0
compression-ratio    0
horsepower           0
peak-rpm             0
city-mpg             0
highway-mpg          0
price                0
dtype: int64
```

In [34]:

```
data.dtypes
```

Out[34]:
```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

In [ ]:

Loading [MathJax]/extensions/Safe.js

In [1]:
```python
# DSA_Experiment_2
# Nandini Pramod Junghare (211091012)
# Astha Shankar Shinde (211091044)


#Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt

tips_data = sns.load_dataset("tips")
```

In [2]:
```python
tips_data.head()
```

Out[2]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

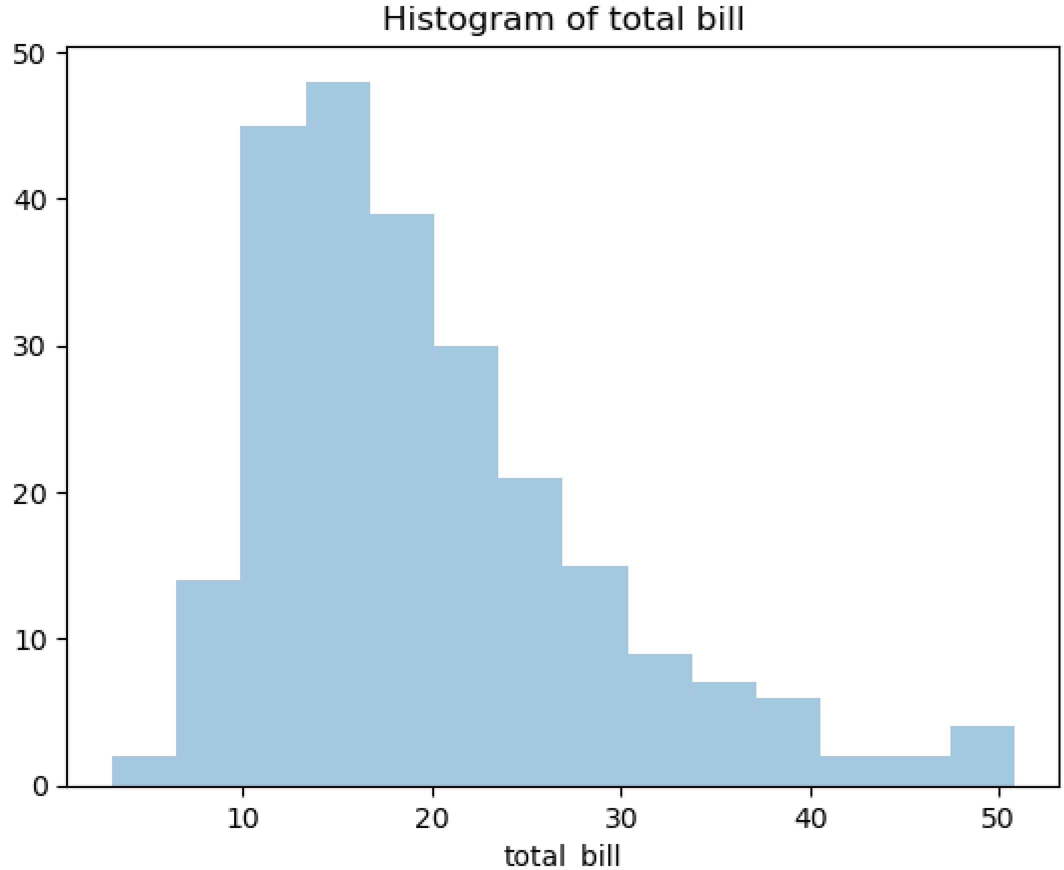
In [3]:
```python
# gives you a summary of the statistical properties of the numerical columns in the data
tips_data.describe()
```

Out[3]:

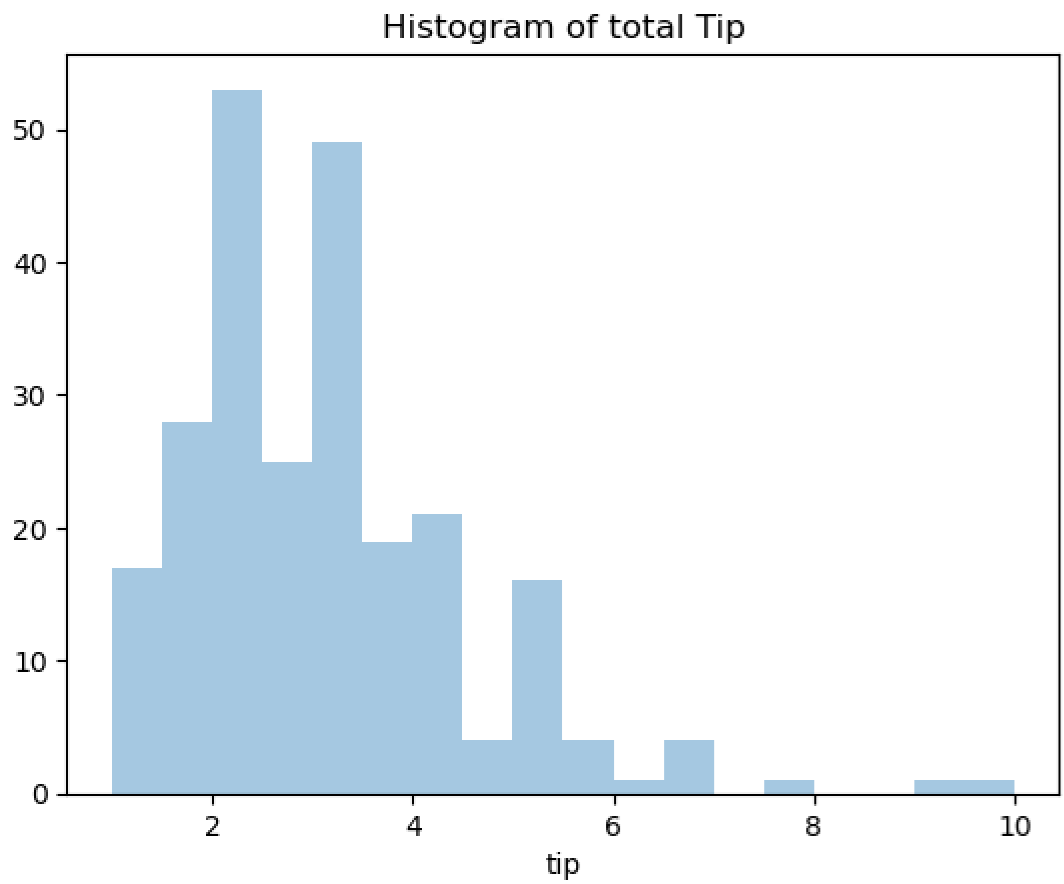
	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

In [4]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# it's creating a visual representation of the distribution of total bills in the datase
tips_data = sns.load_dataset("tips")
sns.distplot(tips_data["total_bill"], kde=False).set_title("Histogram of total bill")
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



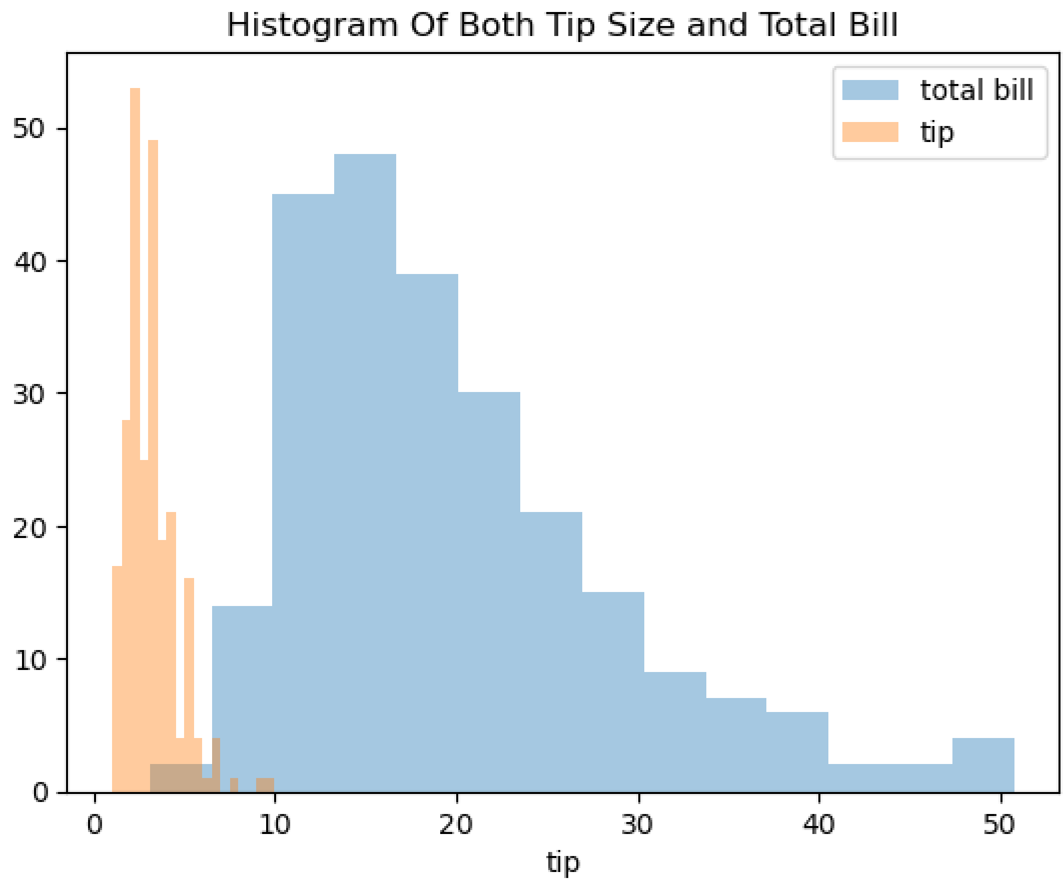
In [5]:
```
sns.distplot(tips_data["tip"], kde= False).set_title("Histogram of total Tip")
plt.show()
```



In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.load_dataset("tips")
```

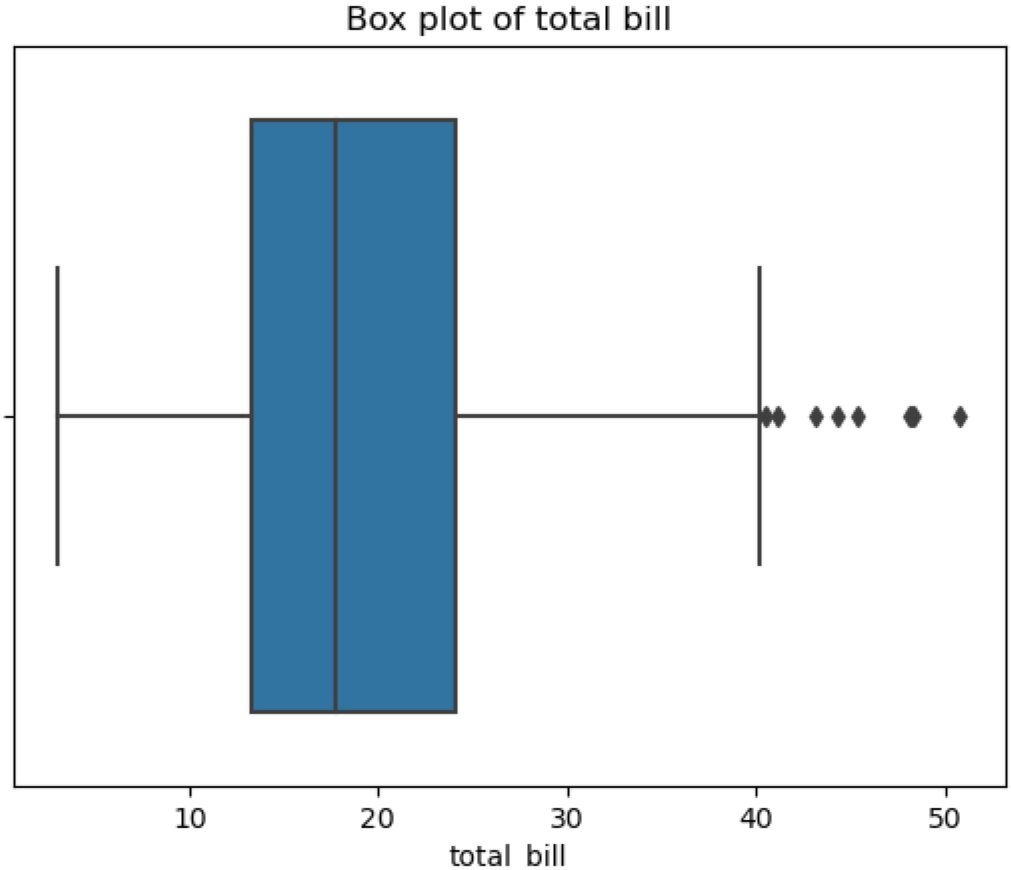
In [7]:
```python
#Histogram of both
sns.distplot(tips_data["total_bill"], kde= False)
sns.distplot(tips_data["tip"], kde=False).set_title("Histogram Of Both Tip Size and Tota
plt.legend(['total bill', 'tip'])
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



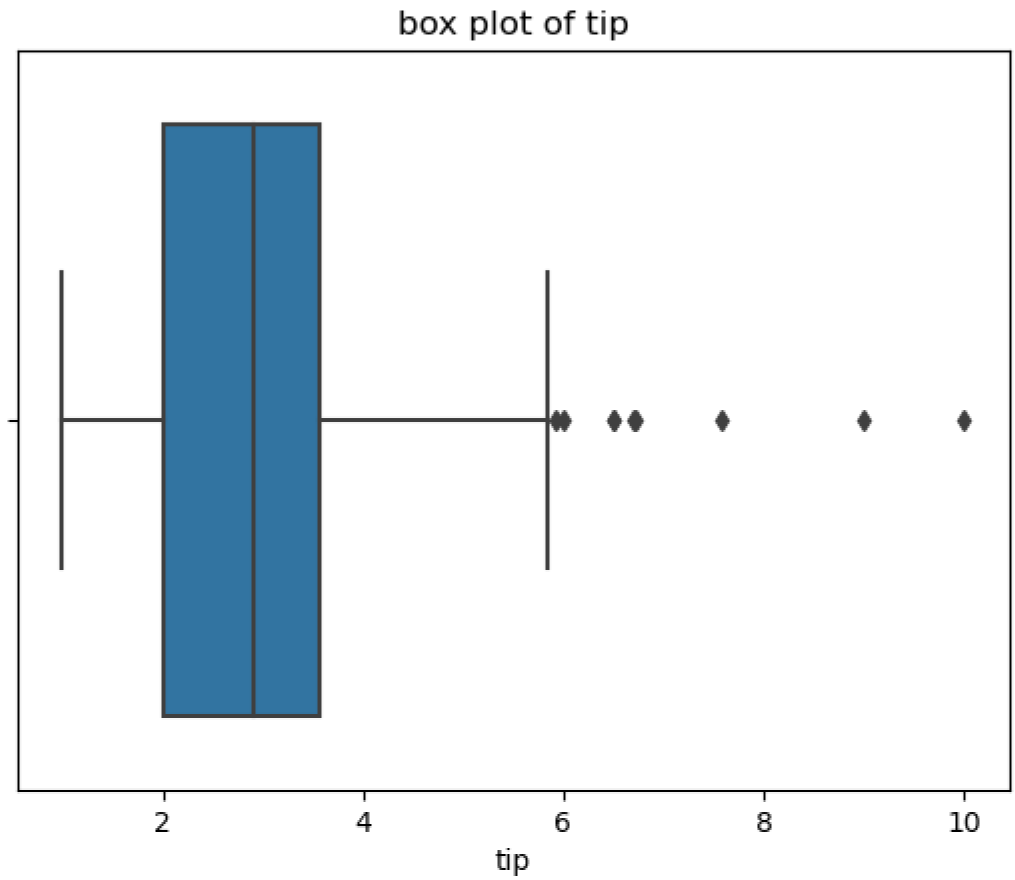
In [8]:
```python
# it's a visual representation of the distribution of total bills using a box plot.
# The box plot consists of a box that spans from Q1 to Q3, with a line representing the
sns.boxplot(tips_data["total_bill"]).set_title("Box plot of total bill")
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

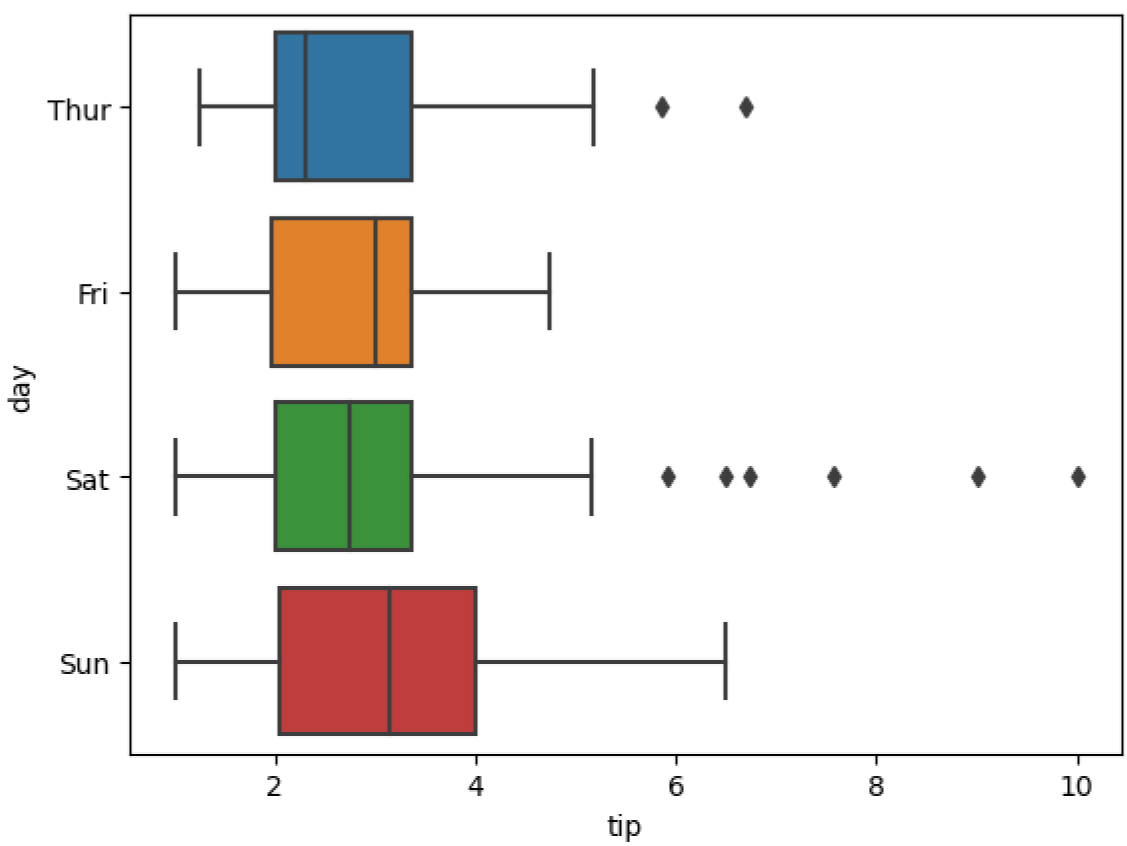


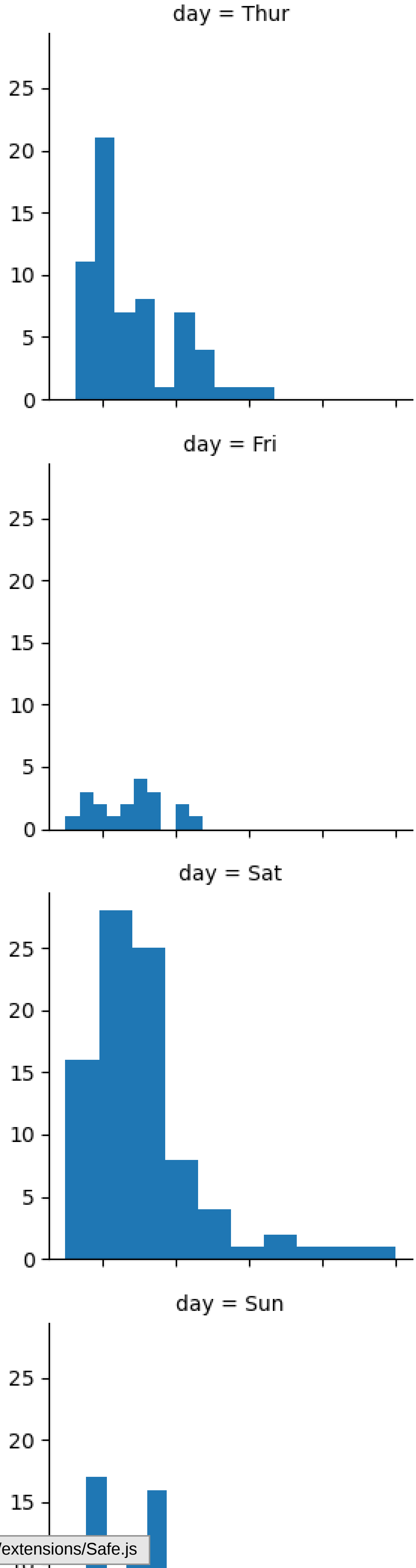
In [9]:
```
sns.boxplot(tips_data["tip"]).set_title("box plot of tip")
plt.show()
```

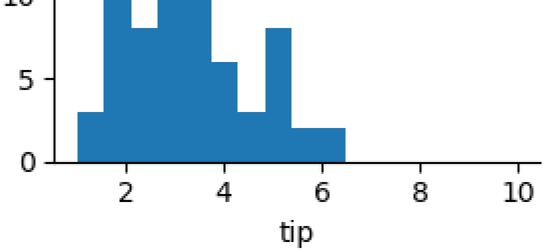
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



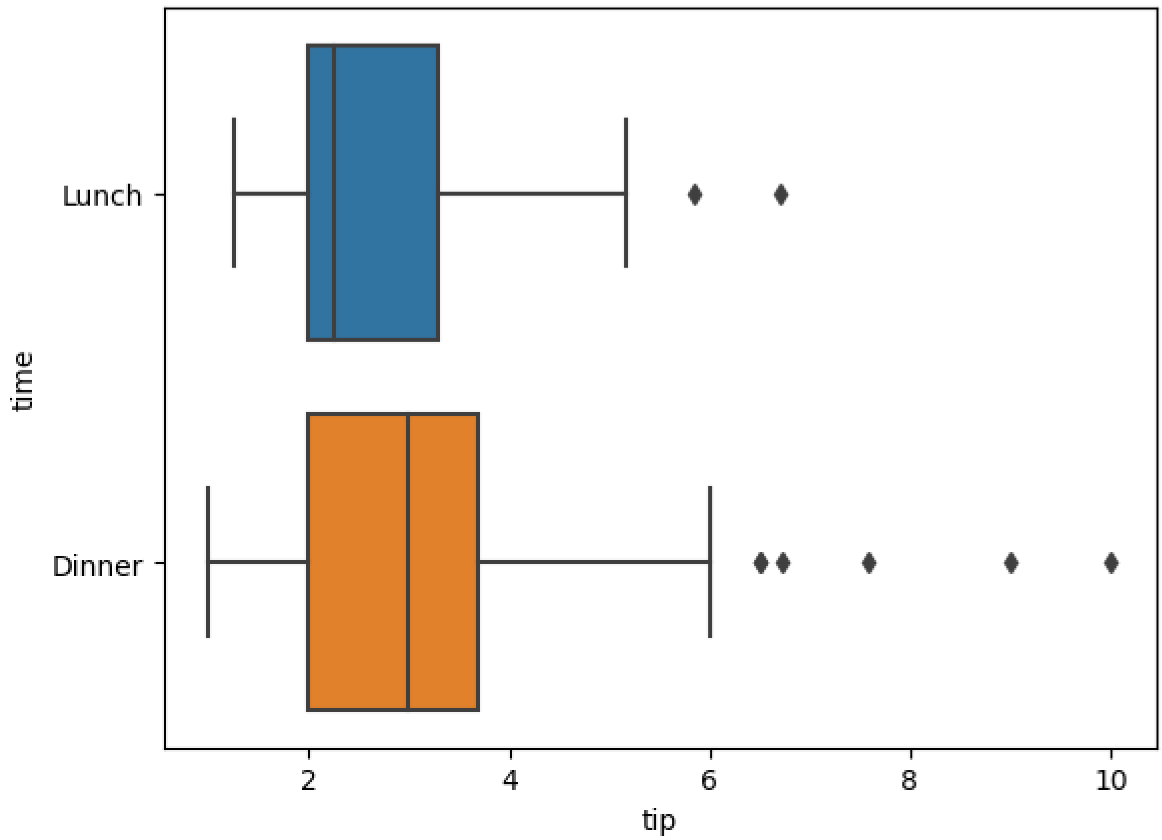
In [14]:
```
import seaborn as sns
sns.boxplot(x=tips_data["tip"],y =tips_data["day"])
plt.show()
g = sns.FacetGrid(tips_data, row="day")
g = g.map(plt.hist, "tip")
plt.show()
```



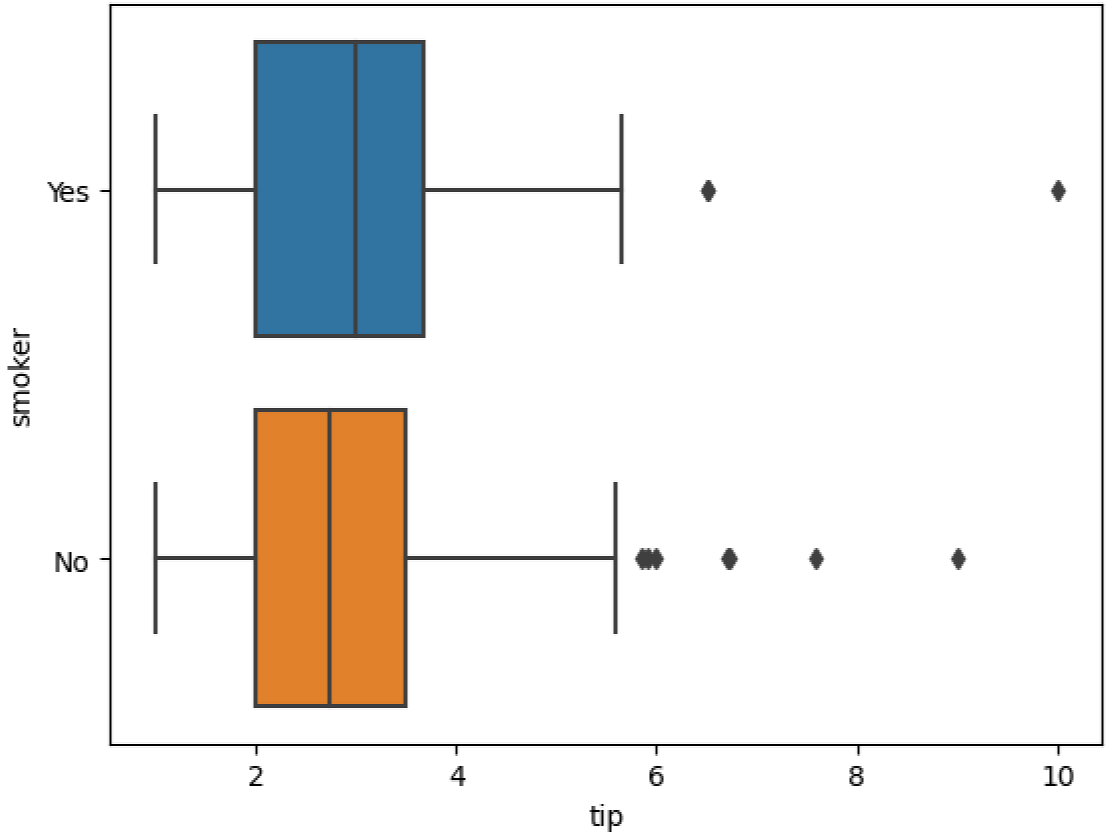

day = Thur
25
20
15
10
5
0
day = Fri
25
20
15
10
5
0
day = Sat
25
20
15
10
5
0
day = Sun
25
20
15



In [11]:
```python
import seaborn as sns
sns.boxplot(x=tips_data["tip"],y =tips_data["time"])
plt.show()
```

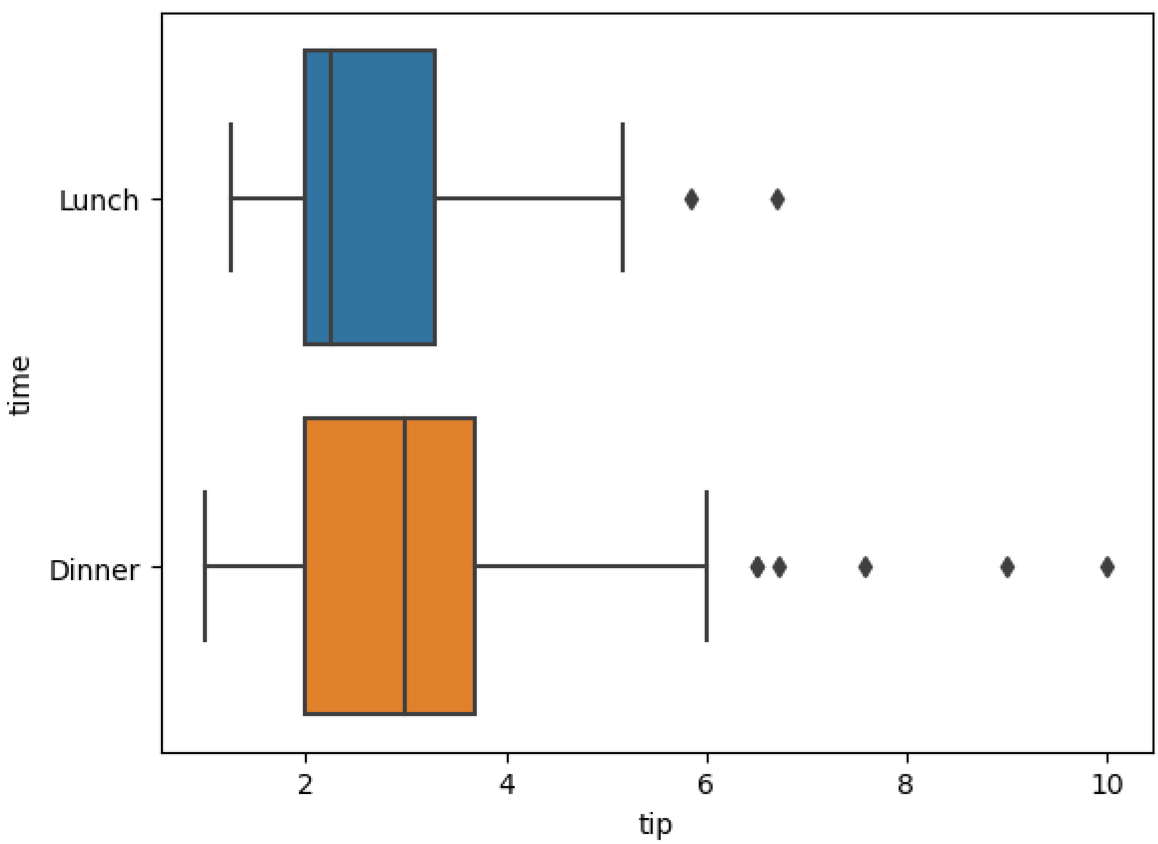


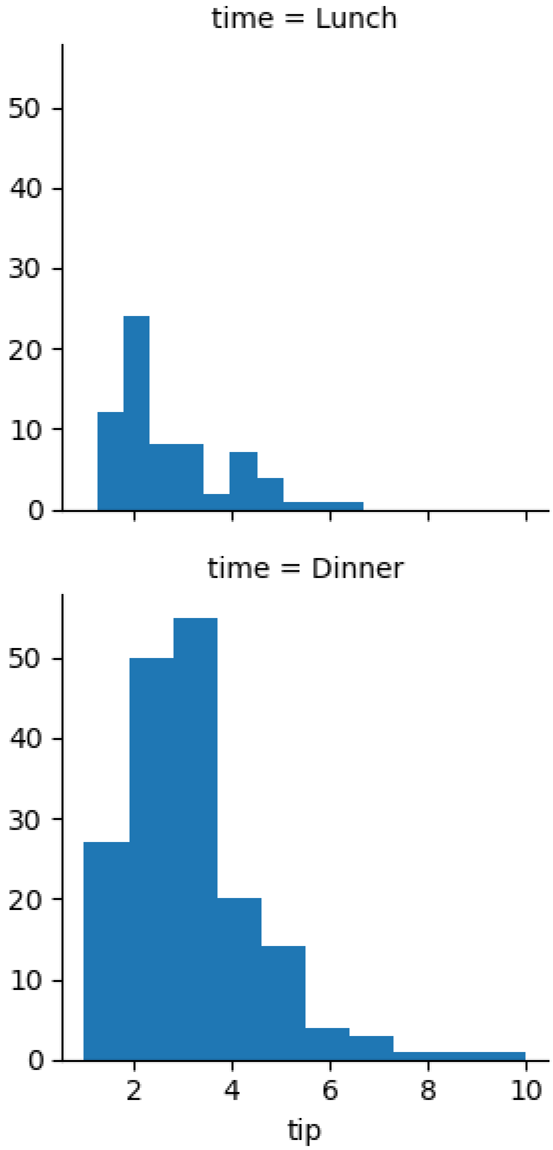
In [12]:
```python
import seaborn as sns
sns.boxplot(x=tips_data["tip"],y =tips_data["smoker"])
plt.show()
```



In [13]:

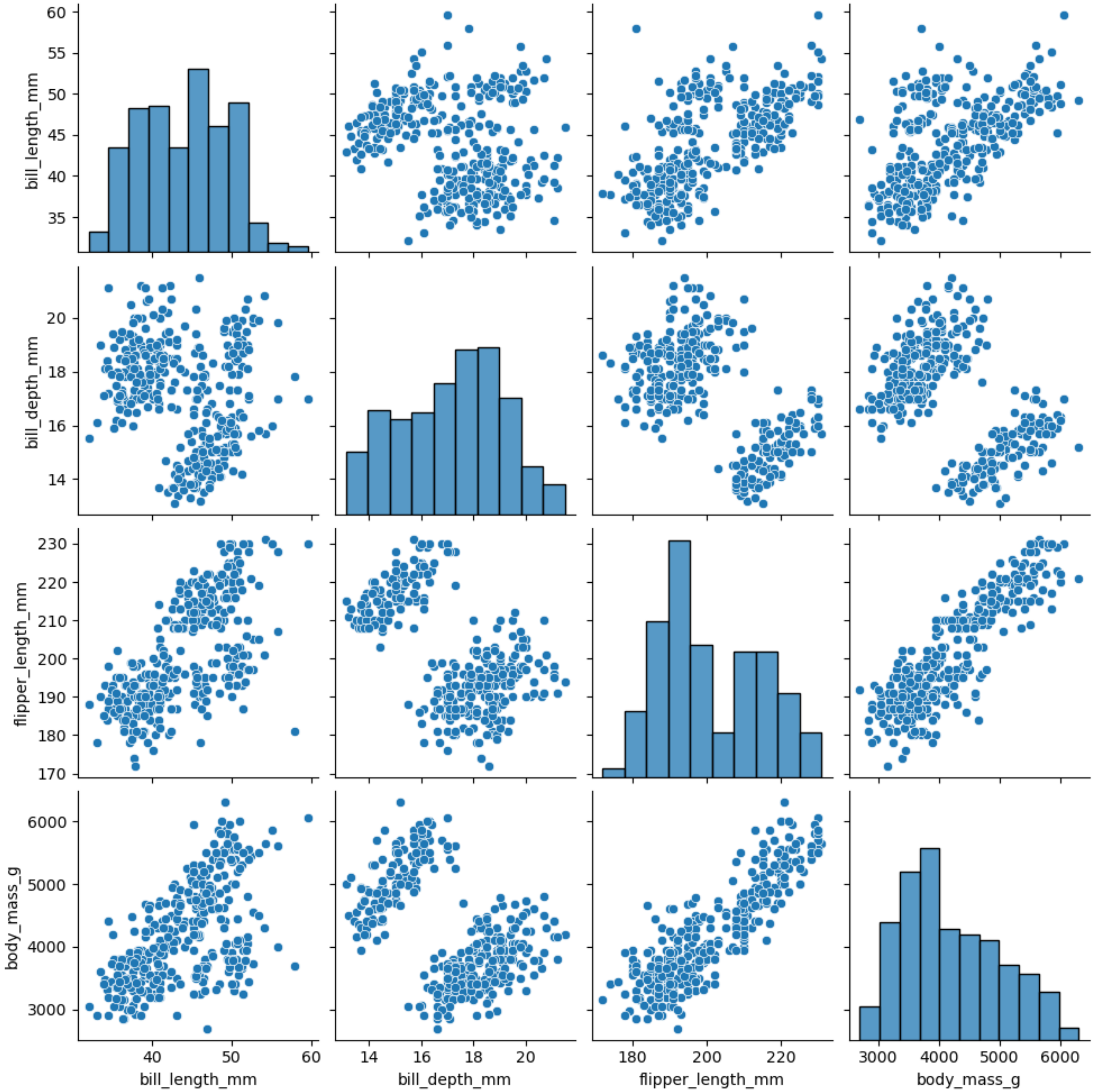
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=tips_data["tip"], y=tips_data["time"])
plt.show()
g = sns.FacetGrid(tips_data, row="time")
g = g.map(plt.hist, "tip")
plt.show()
```





In [16]:
```
penguins = sns.load_dataset("penguins")
sns.pairplot(penguins)
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x1b88d3edb80>
```


bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g
60
55
50
45
40
35
20
18
16
14
230
220
210
200
190
180
170
6000
5000
4000
3000
40
50
60
14
16
18
20
180
200
220
3000
4000
5000
6000
bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g

In [ ]:

Conclusion: We have explored two fundamental aspects of data analysis: data wrangling and visualization. Data wrangling involves identifying and addressing missing values by calculating the average (mean) and replacing null values with this mean. Additionally, data visualization allows us to visually represent data through histograms and box plots. In box plots, we learned about the interquartile range (IQR), which measures the spread of the middle 50% of the data. A larger IQR indicates greater variability in the data. Moreover, the presence of more outliers in a dataset suggests increased variability or dispersion. These concepts are crucial in understanding and interpreting data effectively for various analytical purposes.