Loading the Data

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [2]:
```
ufo_df = pd.read_csv('UFO_data.csv')
ufo_df.head()
```

Out[2]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	54321.0	Willingboro	green	NaN	NJ	6/30/1930 20:00
2	87654.0	Holyoke	NaN	OVAL	CO	2/15/1931 14:00
3	3456.0	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.0	New York Worlds Fair	NaN	LIGHT	NY	4/18/1933 19:00

In [3]:
```
#Printung the size of the data
ufo_df.shape
```

Out[3]:
```
(15, 6)
```

In [4]:
```
ufo_df.dtypes
```

Out[4]:
```
Number              float64
City                 object
Colors Reported      object
Shape Reported       object
State                object
Time                 object
dtype: object
```

Checking for null values Column wise

In [5]:
```
ufo_df.isnull().any()
```

Out[5]:
```
Number               True
City                False
Colors Reported      True
Shape Reported       True
State                True
Time                False
dtype: bool
```

In [6]:
```
ufo_df.isnull()
```

Out[6]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	False	False	True	False	False	False
1	False	False	False	True	False	False
2	False	False	True	False	False	False
3	False	False	False	False	False	False
4	False	False	True	False	False	False
5	True	False	True	True	False	False
6	False	False	False	False	False	False
7	False	False	True	False	False	False
8	True	False	False	False	False	False
9	False	False	False	False	True	False
10	False	False	True	False	False	False
11	True	False	True	False	False	False
12	False	False	False	False	False	False
13	False	False	True	False	False	False
14	False	False	True	False	False	False

Printing the number of null Values in eaach column.

In [7]:
```
print('Total Missing values in the Data :- ',ufo_df.isnull().sum())
```

```
Total Missing values in the Data :-  Number              3
City                0
Colors Reported     9
Shape Reported      2
State               1
Time                0
dtype: int64
```

Printing the total number of null Values in Dataset

In [8]:
```
print('Total Missing values in the Data :- ',ufo_df.isnull().sum().sum())
```

```
Total Missing values in the Data :-  15
```

Missing Value Treatment

1)Dropping Null/Missing Values

In [9]:
```
#Shape of the Data before Dropping missing values
ufo_df.shape
```

Out[9]:
```
(15, 6)
```

In [10]:
```
# Droping the missing or null values
data1 = ufo_df.dropna()
```

In [11]:
```
data1
```

Out[11]:

	Number	City	Colors Reported	Shape Reported	State	Time
3	3456.0	Abilene	blue	DISK	KS	1/12/1900 0:00
6	87654.0	Crater Lake	yellow	CIRCLE	CA	6/15/1935 0:00
12	6543.0	Belton	red	SPHERE	Y	6/30/1939 20:00

In [12]:
```
data1.shape
```

Out[12]:
```
(3, 6)
```

In [13]:
```
print('Loss of the Data : ',round((ufo_df.shape[0]-data1.shape[0])*100/ufo_df.shape[0],2),'%')
```

```
Loss of the Data :  80.0 %
```

We are `losing nearly 80 % data by dropping the missing` value rows which would be a loss for the prediction

`Not a good technique for the Data Analytics`

2) Filling Missing Values

`Filling Missing Values function is used to replace the null values with a test statistic like mean, median or mode of the particular feature`

We can also specify a `forward-fill` or `back-fill` to propagate `the next values backward` or `previous value forward`.

In [14]:
```
data_new = ufo_df
```

In [15]:
```
data_new.isnull().any()
```

Out[15]:
```
Number               True
City                False
Colors Reported      True
Shape Reported       True
State                True
Time                False
dtype: bool
```

Using `Back-fill` or `forward-fill` to propagate next or previous values respectively:

In [16]:
```
## Before applying fill function on the data.
ufo_df.head()
```

Out[16]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	54321.0	Willingboro	green	NaN	NJ	6/30/1930 20:00
2	87654.0	Holyoke	NaN	OVAL	CO	2/15/1931 14:00
3	3456.0	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.0	New York Worlds Fair	NaN	LIGHT	NY	4/18/1933 19:00

In [17]:
```
## After backward-fill
data_new = data_new.fillna(method='bfill')
#for forward-fill
#train.fillna(method=''ffill)
data_new
```

Out[17]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.0	Ithaca	green	TRIANGLE	NY	6/1/1930 22:00
1	54321.0	Willingboro	green	OVAL	NJ	6/30/1930 20:00
2	87654.0	Holyoke	blue	OVAL	CO	2/15/1931 14:00
3	3456.0	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.0	New York Worlds Fair	yellow	LIGHT	NY	4/18/1933 19:00
5	87654.0	Valley City	yellow	CIRCLE	N	9/15/1934 15:30
6	87654.0	Crater Lake	yellow	CIRCLE	CA	6/15/1935 0:00
7	6789.0	Alma	green	DISK	MI	7/15/1936 0:00
8	4567.0	Eklutna	green	12	AK	10/15/1936 17:00
9	4567.0	Hubbard	red	CYLINDER	CA	6/15/1937 0:00
10	7654.0	Fontana	red	LIGHT	CA	8/15/1937 21:00
11	6543.0	Waterloo	red	FIREBALL	AL	6/1/1939 20:00
12	6543.0	Belton	red	SPHERE	Y	6/30/1939 20:00
13	876.0	Keokuk	NaN	@	IA	7/7/1939 2:00
14	5432.0	Ludington	NaN	DISK	MI	6/1/1941 13:00

In [18]:
```
# After forward-fill
data_new = data_new.fillna(method='ffill')
data_new
```

Out[18]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.0	Ithaca	green	TRIANGLE	NY	6/1/1930 22:00
1	54321.0	Willingboro	green	OVAL	NJ	6/30/1930 20:00
2	87654.0	Holyoke	blue	OVAL	CO	2/15/1931 14:00
3	3456.0	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.0	New York Worlds Fair	yellow	LIGHT	NY	4/18/1933 19:00
5	87654.0	Valley City	yellow	CIRCLE	N	9/15/1934 15:30
6	87654.0	Crater Lake	yellow	CIRCLE	CA	6/15/1935 0:00
7	6789.0	Alma	green	DISK	MI	7/15/1936 0:00
8	4567.0	Eklutna	green	12	AK	10/15/1936 17:00
9	4567.0	Hubbard	red	CYLINDER	CA	6/15/1937 0:00
10	7654.0	Fontana	red	LIGHT	CA	8/15/1937 21:00
11	6543.0	Waterloo	red	FIREBALL	AL	6/1/1939 20:00
12	6543.0	Belton	red	SPHERE	Y	6/30/1939 20:00
13	876.0	Keokuk	red	@	IA	7/7/1939 2:00
14	5432.0	Ludington	red	DISK	MI	6/1/1941 13:00

What Happens if for a missing value the Next value or previous value is also a Nan

If a previous or next value isn't available or rather if it is also a NaN value, then, the NaN remains even after back-filling or forward-filling.

In [19]:
```
data_new.isnull().any()
```

Out[19]:
```
Number              False
City                False
Colors Reported     False
Shape Reported      False
State               False
Time                False
dtype: bool
```

Here one columns are still having Missing values

Filling the missing value with a statistic

In [20]:
```
data_new = ufo_df
```

In [21]:
```
# 'Number' is a column name for our data
mean_value = data_new['Number'].mean()
print("Mean Value", mean_value)
#this will replace all NaN values with the mean of the non null values
data_new['Number']=data_new['Number'].fillna(mean_value)
data_new.head()
```

```
Mean Value 33004.666666666664
```

Out[21]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	54321.0	Willingboro	green	NaN	NJ	6/30/1930 20:00
2	87654.0	Holyoke	NaN	OVAL	CO	2/15/1931 14:00
3	3456.0	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.0	New York Worlds Fair	NaN	LIGHT	NY	4/18/1933 19:00

In [22]:
```
data_new
```

Out[22]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.000000	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	54321.000000	Willingboro	green	NaN	NJ	6/30/1930 20:00
2	87654.000000	Holyoke	NaN	OVAL	CO	2/15/1931 14:00
3	3456.000000	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.000000	New York Worlds Fair	NaN	LIGHT	NY	4/18/1933 19:00
5	33004.666667	Valley City	NaN	NaN	N	9/15/1934 15:30
6	87654.000000	Crater Lake	yellow	CIRCLE	CA	6/15/1935 0:00
7	6789.000000	Alma	NaN	DISK	MI	7/15/1936 0:00
8	33004.666667	Eklutna	green	12	AK	10/15/1936 17:00
9	4567.000000	Hubbard	red	CYLINDER	NaN	6/15/1937 0:00
10	7654.000000	Fontana	NaN	LIGHT	CA	8/15/1937 21:00
11	33004.666667	Waterloo	NaN	FIREBALL	AL	6/1/1939 20:00
12	6543.000000	Belton	red	SPHERE	Y	6/30/1939 20:00
13	876.000000	Keokuk	NaN	@	IA	7/7/1939 2:00
14	5432.000000	Ludington	NaN	DISK	MI	6/1/1941 13:00

In [23]:
```
data_new.isnull().any()
```

Out[23]:
```
Number              False
City                False
Colors Reported      True
Shape Reported       True
State                True
Time                False
dtype: bool
```

In [29]:
```
data_new = ufo_df
```

In [30]:
```
#tip is a column name for our data
median_value = data_new['Number'].median
print("median Value: ",mean_value )
#this will replace all NaN values with the mean of the non null values
data_new['Number'] = data_new['Number'].fillna(median_value)
```

```
median Value:  33004.666666666664
```

Similarly the missing values can be filled by using the `median method`

In [31]:
```
data_new.head()
```

Out[31]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.0	Ithaca	NaN	TRIANGLE	NY	6/1/1930 22:00
1	54321.0	Willingboro	green	NaN	NJ	6/30/1930 20:00
2	87654.0	Holyoke	NaN	OVAL	CO	2/15/1931 14:00
3	3456.0	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.0	New York Worlds Fair	NaN	LIGHT	NY	4/18/1933 19:00

In [32]:
```
#Filling the categorical variables with mode method of imputation
mode_value = data_new['Colors Reported'].mode()
mode_value[0]
```

Out[32]:
```
'green'
```

In [33]:
```
data_new['Colors Reported'] = data_new['Colors Reported'].fillna(mode_value[0])
```

In [34]:
```
data_new
```

Out[34]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.000000	Ithaca	green	TRIANGLE	NY	6/1/1930 22:00
1	54321.000000	Willingboro	green	NaN	NJ	6/30/1930 20:00
2	87654.000000	Holyoke	green	OVAL	CO	2/15/1931 14:00
3	3456.000000	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.000000	New York Worlds Fair	green	LIGHT	NY	4/18/1933 19:00
5	33004.666667	Valley City	green	NaN	N	9/15/1934 15:30
6	87654.000000	Crater Lake	yellow	CIRCLE	CA	6/15/1935 0:00
7	6789.000000	Alma	green	DISK	MI	7/15/1936 0:00
8	33004.666667	Eklutna	green	12	AK	10/15/1936 17:00
9	4567.000000	Hubbard	red	CYLINDER	NaN	6/15/1937 0:00
10	7654.000000	Fontana	green	LIGHT	CA	8/15/1937 21:00
11	33004.666667	Waterloo	green	FIREBALL	AL	6/1/1939 20:00
12	6543.000000	Belton	red	SPHERE	Y	6/30/1939 20:00
13	876.000000	Keokuk	green	@	IA	7/7/1939 2:00
14	5432.000000	Ludington	green	DISK	MI	6/1/1941 13:00

In [35]:
```
mode_value = data_new['Shape Reported'].mode()
data_new['Shape Reported'] = data_new['Shape Reported'].fillna(mode_value[0])


data_new
```

Out[35]:

	Number	City	Colors Reported	Shape Reported	State	Time
0	123456.000000	Ithaca	green	TRIANGLE	NY	6/1/1930 22:00
1	54321.000000	Willingboro	green	DISK	NJ	6/30/1930 20:00
2	87654.000000	Holyoke	green	OVAL	CO	2/15/1931 14:00
3	3456.000000	Abilene	blue	DISK	KS	1/12/1900 0:00
4	7654.000000	New York Worlds Fair	green	LIGHT	NY	4/18/1933 19:00
5	33004.666667	Valley City	green	DISK	N	9/15/1934 15:30
6	87654.000000	Crater Lake	yellow	CIRCLE	CA	6/15/1935 0:00
7	6789.000000	Alma	green	DISK	MI	7/15/1936 0:00
8	33004.666667	Eklutna	green	12	AK	10/15/1936 17:00
9	4567.000000	Hubbard	red	CYLINDER	NaN	6/15/1937 0:00
10	7654.000000	Fontana	green	LIGHT	CA	8/15/1937 21:00
11	33004.666667	Waterloo	green	FIREBALL	AL	6/1/1939 20:00
12	6543.000000	Belton	red	SPHERE	Y	6/30/1939 20:00
13	876.000000	Keokuk	green	@	IA	7/7/1939 2:00
14	5432.000000	Ludington	green	DISK	MI	6/1/1941 13:00

In [36]:
```
data_new.isnull().any()
```

Out[36]:
```
Number              False
City                False
Colors Reported     False
Shape Reported      False
State                True
Time                False
dtype: bool
```

Observations

By mean value imputation for numerical data type and mode value imputation for categorical data ,we are able to completely fill the null values,but is it a effective method.

Let's find out with a example

Let x be list of age of people who are attending a Gym in a Office

Data without Outlier's

x = [20,25,25,26,28,29,30,35,38,40,45,30]

mean(x) = 31

Data with missing values and a single Outlier

x = [20, ,*26,* ,28,29,30, ,*35,38,* ,40,100]

mean(x) = 38

We can observe how much the mean value is deviated due to single outlier.So mean is ineffective in presence of outlier's

So ,If outlier's are present in the data then median is the best out of box method

Which method would be most suitable for Imputing the missing values in Class Lables

Mode method of imputation would work but by that we may introduce bias into the model to avoid this ,we generally move those datapoints into test set and try to predict with model

Outlier Treatment

Outlier Detection

In [38]:
```
import pandas as pd
import numpy as np

data = sns.load_dataset("Tips")
data.head()
```

Out[38]:

	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 [39]:
```
data.dtypes
```

Out[39]:
```
total_bill    float64
tip           float64
sex            object
smoker         object
day            object
time           object
size            int64
dtype: object
```

In [40]:
```
#Seprating the continues variables
import matplotlib.pyplot as plt
import seaborn as sns

data2 = data.select_dtypes(exclude = 'object')
data2.head()
```

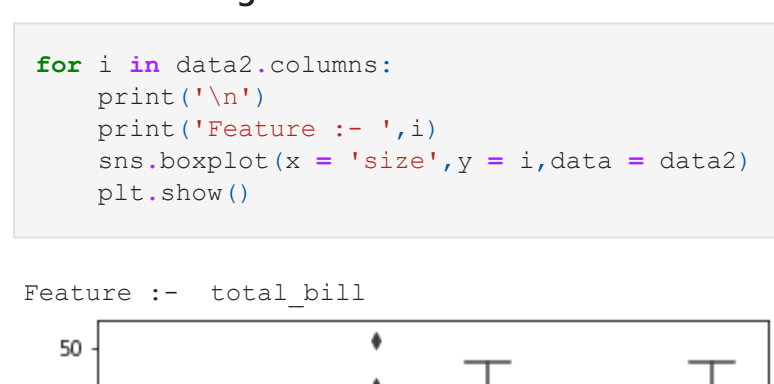
Out[40]:

	total_bill	tip	size
0	16.99	1.01	2
1	10.34	1.66	3
2	21.01	3.50	3
3	23.68	3.31	2
4	24.59	3.61	4

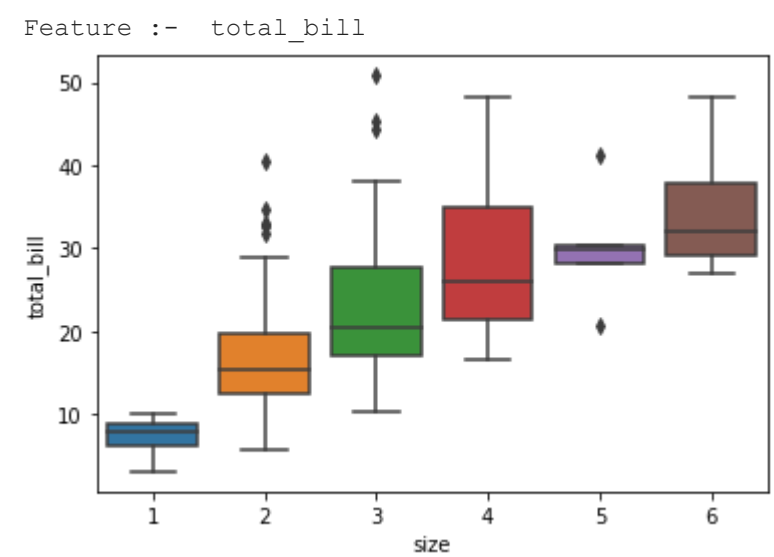
Detection Using Box Plot

In [41]:
```
for i in data2.columns:
    print('\n')
    print('Feature :- ',i)
    sns.boxplot(x = 'size',y = i,data = data2)
    plt.show()
```

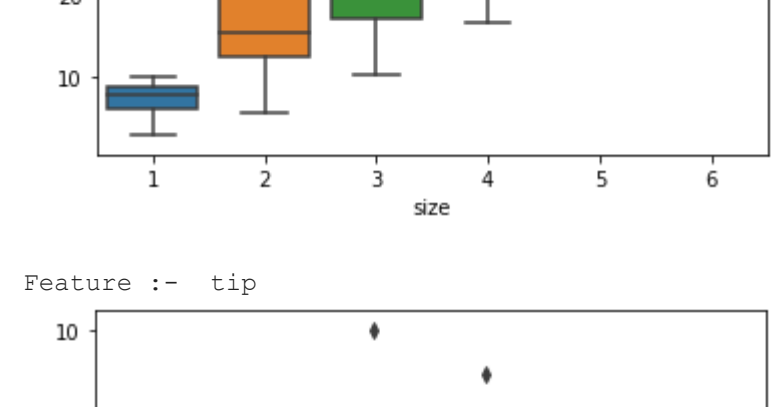
```
Feature :-  total_bill
```




```
Feature :-  tip
```




```
Feature :-  size
```

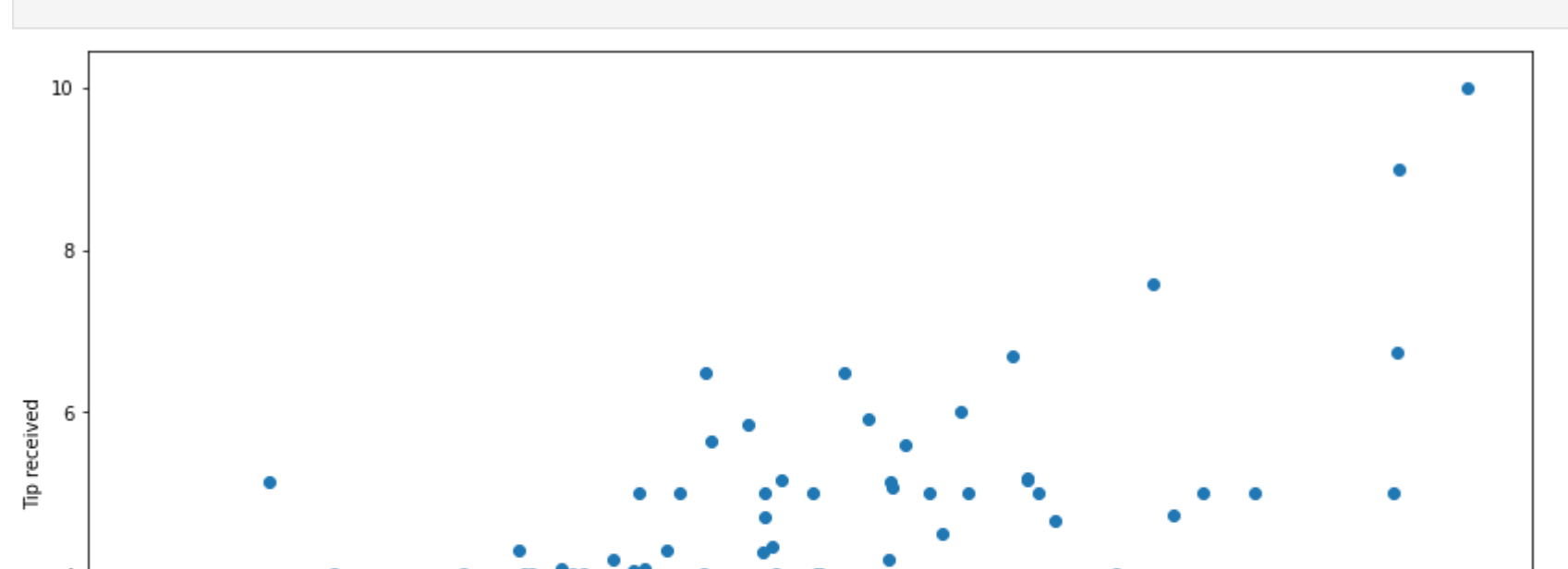


The dots in the box plots of the total bill and tips represent the distribution of the outliers class wise in the data.

The presence of the outlier can severely affect the model performance and may lead to improper conclusions. For example you want to caluculate the total turn over of your restaurent for the next month financial planning.If you caluculate based on outlier data you entire planning may collapse.

Detection using Scatter Plot

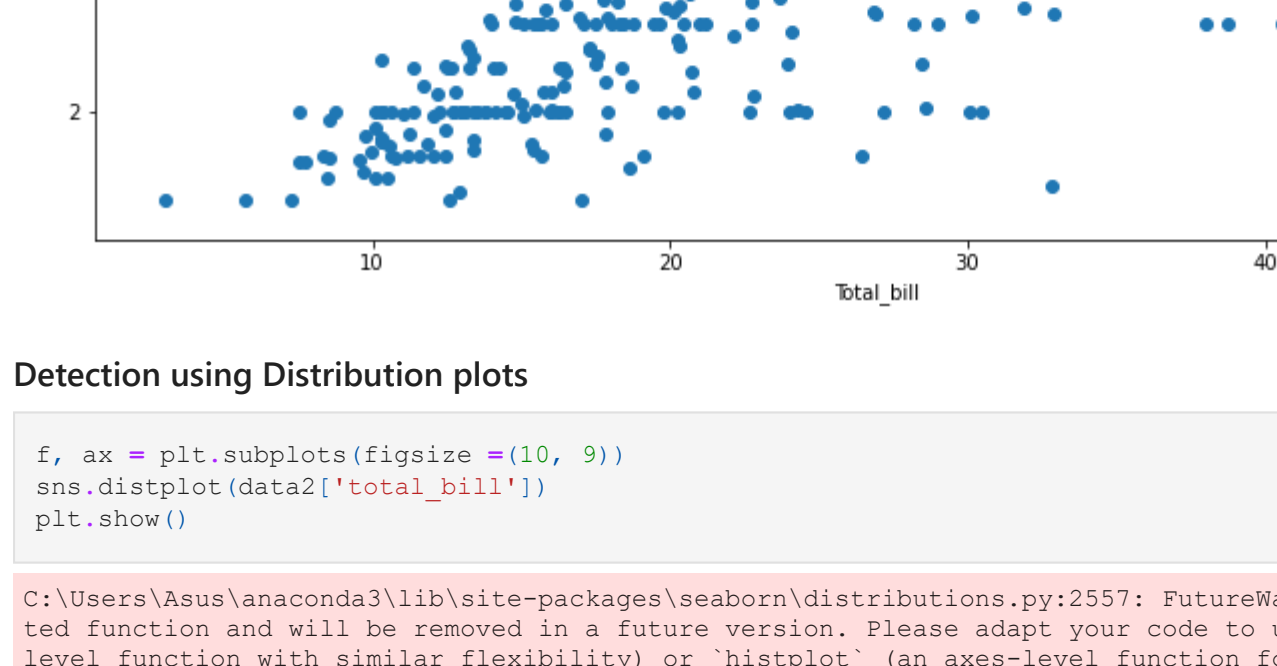
In [42]:
```
fig, ax = plt.subplots(figsize=(14,8))
ax.scatter(data2['total_bill'], data2['tip'])
ax.set_xlabel('Total_bill')
ax.set_ylabel('Tip received')
plt.show()
```



Detection using Distribution plots

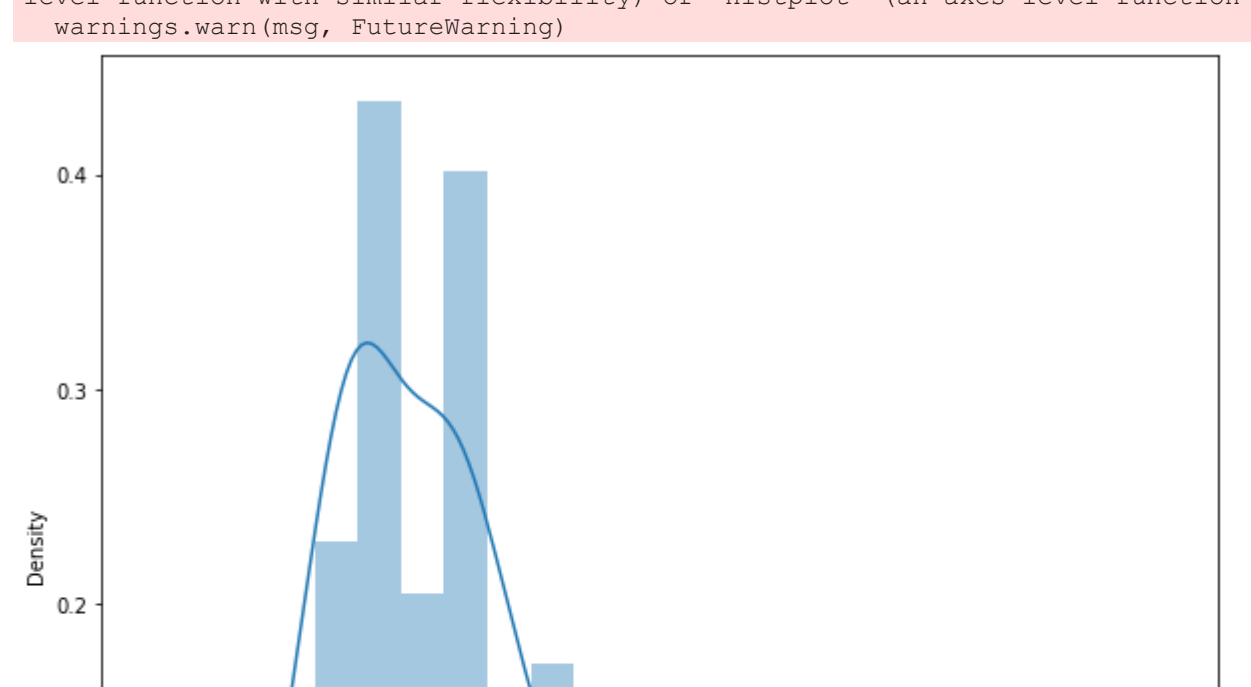
In [43]:
```
f, ax = plt.subplots(figsize =(10, 9))
sns.distplot(data2['total_bill'])
plt.show()
```

```
C:\Users\Asus\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [44]:
```
f, ax = plt.subplots(figsize =(10, 9))
sns.distplot(data2['tip'])
plt.show()
```

```
C:\Users\Asus\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Both the graphs are right skewed due to the presence of outliers.

Outlier Correction

1) Dropping Outlier Points

As discussed above if we drop the outlier points we may face a huge loss of Data.

2)Rounding the Outliers

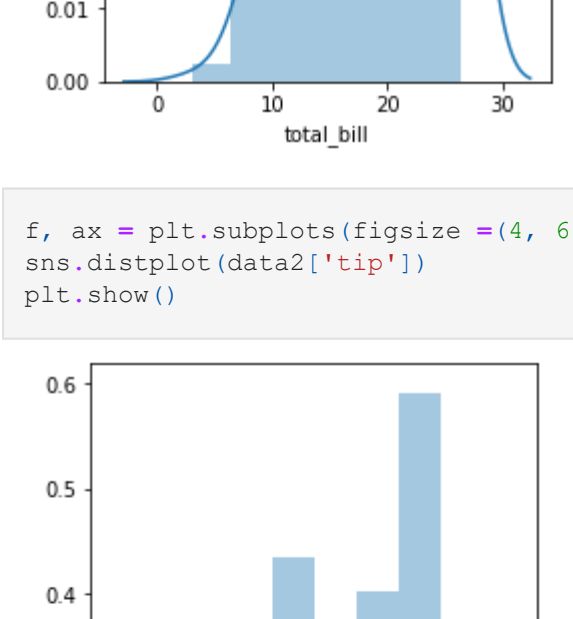
Here we round off all the outlier value with the nearest best percentile value

In [45]:
```
# Winsorize will round off the outlier values to the percentile of the variable that we mention in the limits
from scipy.stats.mstats import winsorize

for column in data2.columns:
    data2[column] = winsorize(data2[column], limits=(0, 0.2), inclusive=(True, True), inplace=False, axis=None)
```

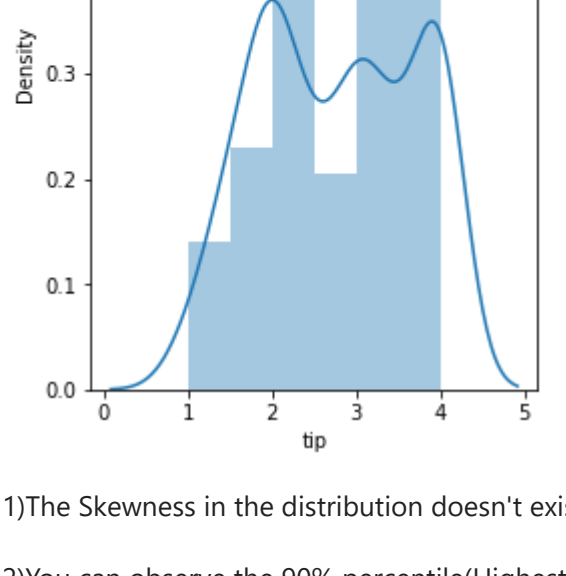
```
<ipython-input-45-1f3004034b66>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data2[column] = winsorize(data2[column], limits=(0, 0.2), inclusive=(True, True), inplace=False, axis=None)
```

Distribution after Outlier Removal

In [46]:
```
import warnings
warnings.filterwarnings('ignore')
f, ax = plt.subplots(figsize =(4, 6))
sns.distplot(data2['total_bill'])
plt.show()
```



In [47]:
```
f, ax = plt.subplots(figsize =(4, 6))
sns.distplot(data2['tip'])
plt.show()
```



1)The Skewness in the distribution doesn't exist anymore as the outliers has been rounded off.

2)You can observe the 90% percentile(Highest bar in the last) going high than before as all outliers are rounded to it.

In [ ]: