Data_visualization using_Seaborn

August 19, 2022

[1]:
```
#data Visualisation using Seaborn
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
#inbuilt Dataset
sns.get_dataset_names()
```

[3]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'taxis',
 'tips',
 'titanic']
```

[4]:
```
#load dataset
data1=sns.load_dataset('tips')
data1
```

[4]:

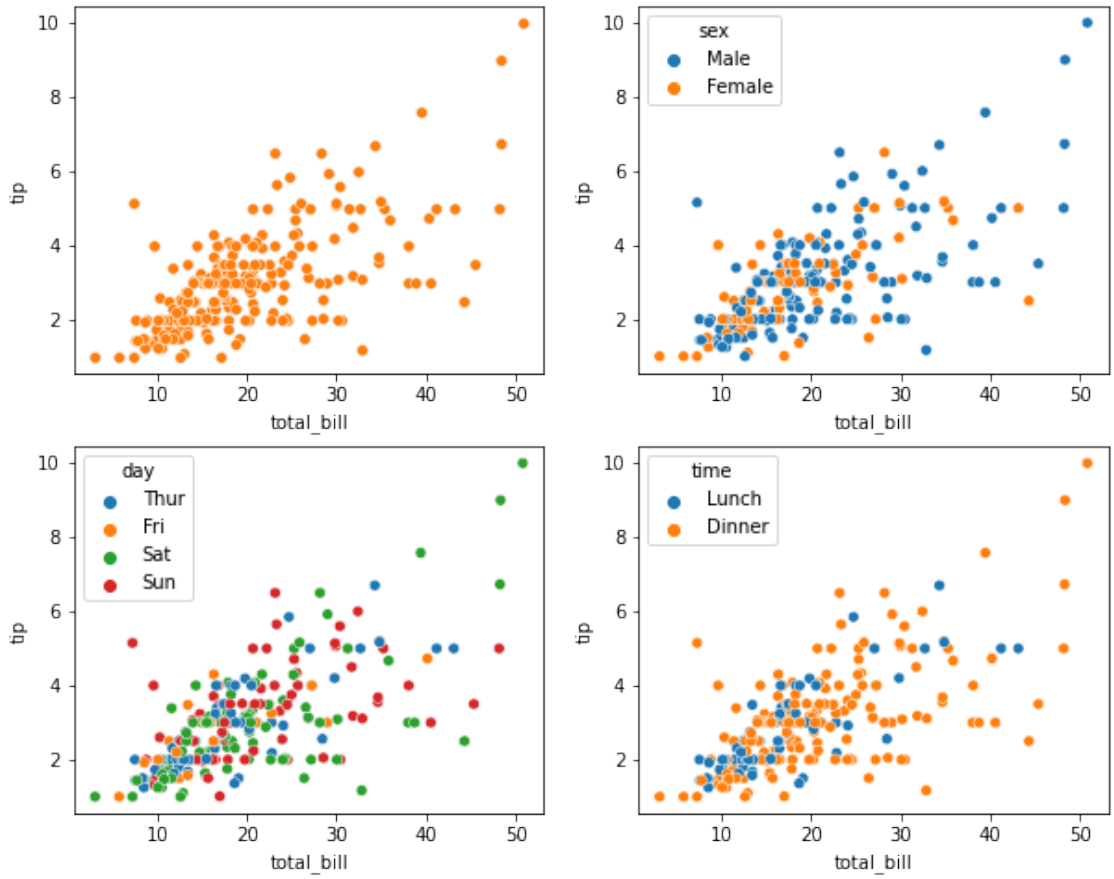
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3

```
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
..           …    …       …      …    …      …    …
239        29.03  5.92    Male     No   Sat  Dinner     3
240        27.18  2.00  Female    Yes   Sat  Dinner     2
241        22.67  2.00    Male    Yes   Sat  Dinner     2
242        17.82  1.75    Male     No   Sat  Dinner     2
243        18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]
```

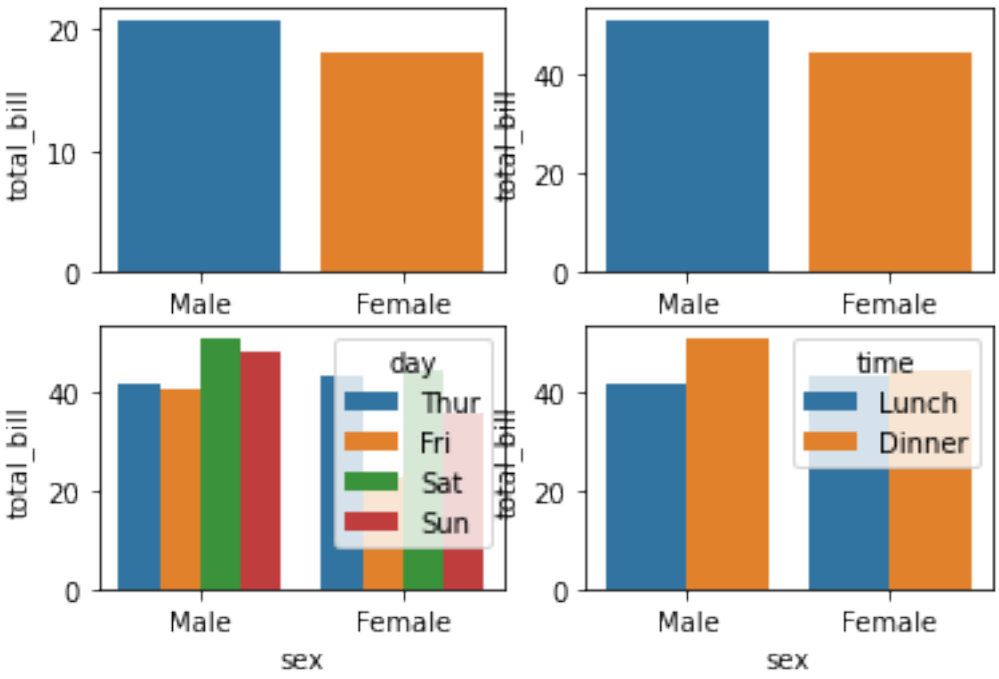
[5]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.3 KB
```

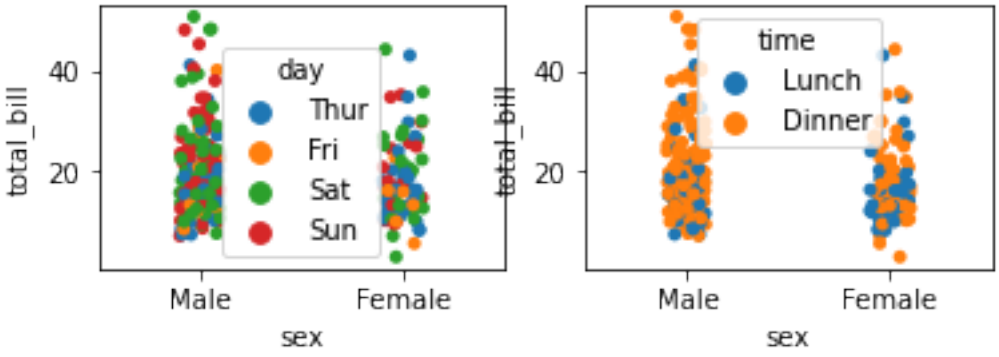
[6]:
```
#scatter plot------find relation btw numerical datas

#find insights btw total bill and tips
plt.figure(figsize=(10,8))
plt.subplot(2,2,1)
sns.scatterplot(x='total_bill',y='tip',data=data1)
sns.scatterplot(x=data1['total_bill'],y=data1['tip'])  # both the syntax return␣
 ↪same thing
plt.subplot(2,2,2)
sns.scatterplot(x=data1['total_bill'],y=data1['tip'],hue=data1['sex'])
plt.subplot(2,2,3)
sns.scatterplot(x=data1['total_bill'],y=data1['tip'],hue=data1['day'])
plt.subplot(2,2,4)
sns.scatterplot(x=data1['total_bill'],y=data1['tip'],hue=data1['time'])
plt.show()
```



[7]:
```
#bar plot------numerical with categorical data
plt.subplot(2,2,1)
sns.barplot(x=data1['sex'],y=data1['total_bill'],ci=None)
plt.subplot(2,2,2)
sns.barplot(x=data1['sex'],y=data1['total_bill'],estimator=np.max,ci=None)
plt.subplot(2,2,3)
sns.barplot(x=data1['sex'],y=data1['total_bill'],estimator=np.
 ↪max,ci=None,hue=data1['day'])
plt.subplot(2,2,4)
sns.barplot(x=data1['sex'],y=data1['total_bill'],estimator=np.
 ↪max,ci=None,hue=data1['time'])
plt.show()
```

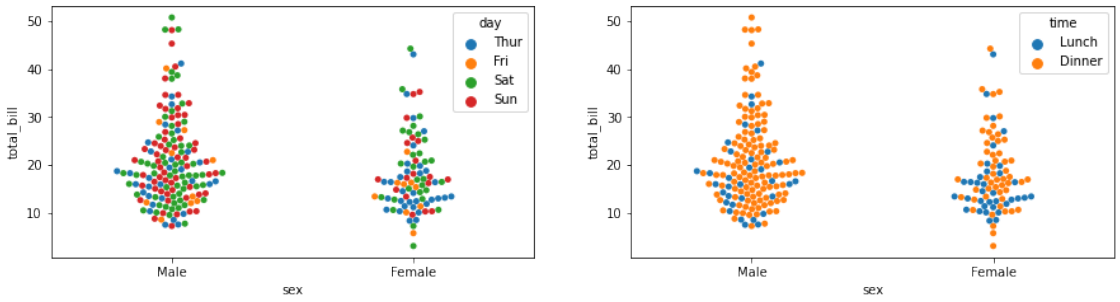


[8]:
```
#stripplot   ---Overlapping happens for almost similar data
plt.subplot(2,2,3)
sns.stripplot(x=data1['sex'],y=data1['total_bill'],hue=data1['day'])
plt.subplot(2,2,4)
sns.stripplot(x=data1['sex'],y=data1['total_bill'],hue=data1['time'])
plt.show()
```

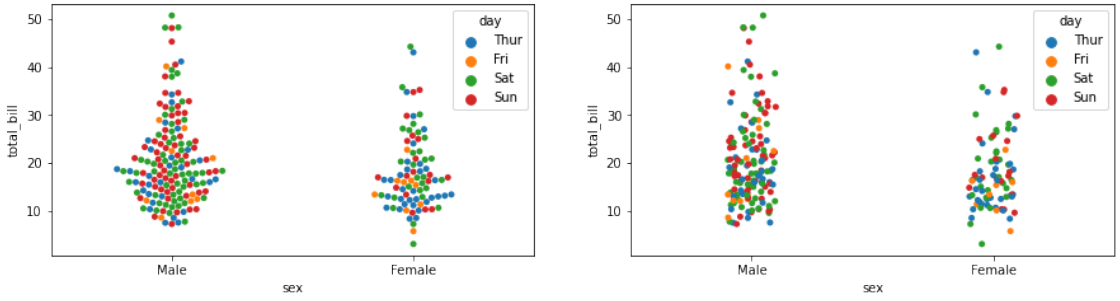


[9]:
```
#swarmpplot   ---Overlapping happens for almost similar data
plt.figure(figsize=(15,8))
plt.subplot(2,2,3)
sns.swarmplot(x=data1['sex'],y=data1['total_bill'],hue=data1['day'])
```

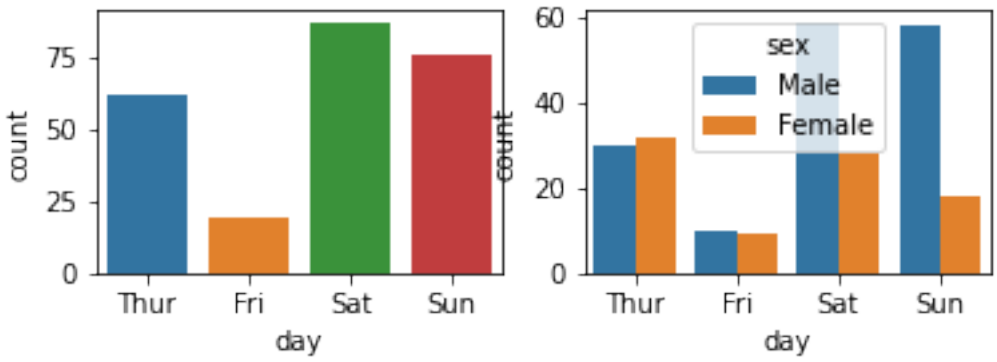
```
plt.subplot(2,2,4)
sns.swarmplot(x=data1['sex'],y=data1['total_bill'],hue=data1['time'])
plt.show()
```



[10]:
```
#diff btw swarm and strip
plt.figure(figsize=(15,8))
plt.subplot(2,2,3)
sns.swarmplot(x=data1['sex'],y=data1['total_bill'],hue=data1['day']) # no␣
↪overlap btw 10 and 10.1 both are represented in diff points
plt.subplot(2,2,4)
sns.stripplot(x=data1['sex'],y=data1['total_bill'],hue=data1['day']) #overlap␣
↪happens btw 10 and 10.1
plt.show()
```



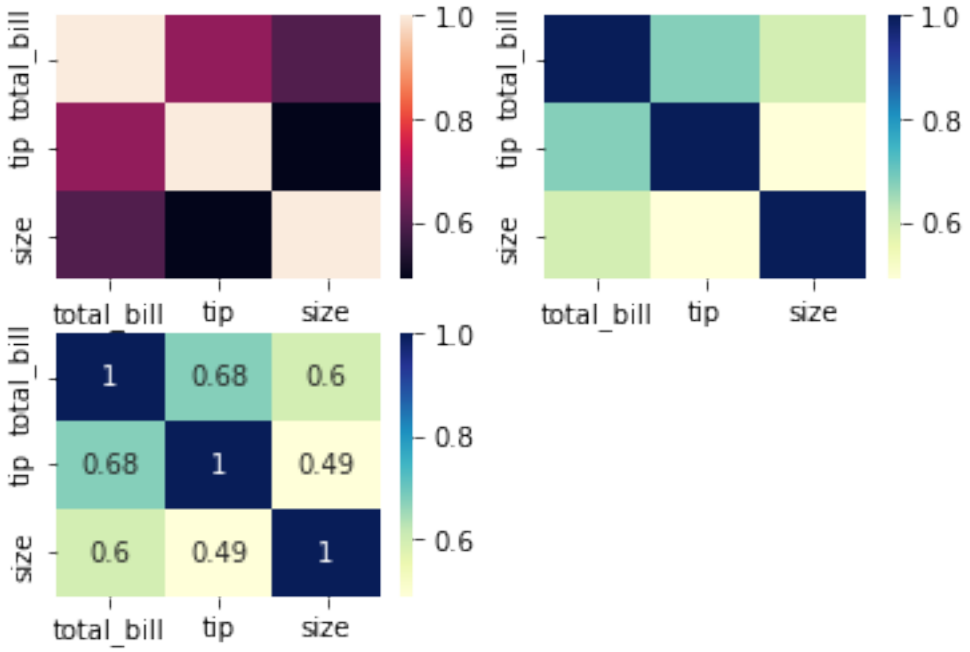
[11]:
```
#countplot
plt.subplot(2,2,1)
sns.countplot(x=data1['day'])
plt.subplot(2,2,2)
sns.countplot(x=data1['day'],hue=data1['sex'])
plt.show()
```



[12]:
```
#heatplot-----Correlation analysis
print(data1.corr()) #------to get correlation between features␣
 ↪default---spearman corr
plt.subplot(2,2,1)
sns.heatmap(data1.corr())
plt.subplot(2,2,2)
sns.heatmap(data1.corr(),cmap='YlGnBu')
plt.subplot(2,2,3)
sns.heatmap(data1.corr(),cmap='YlGnBu',annot=True)
```

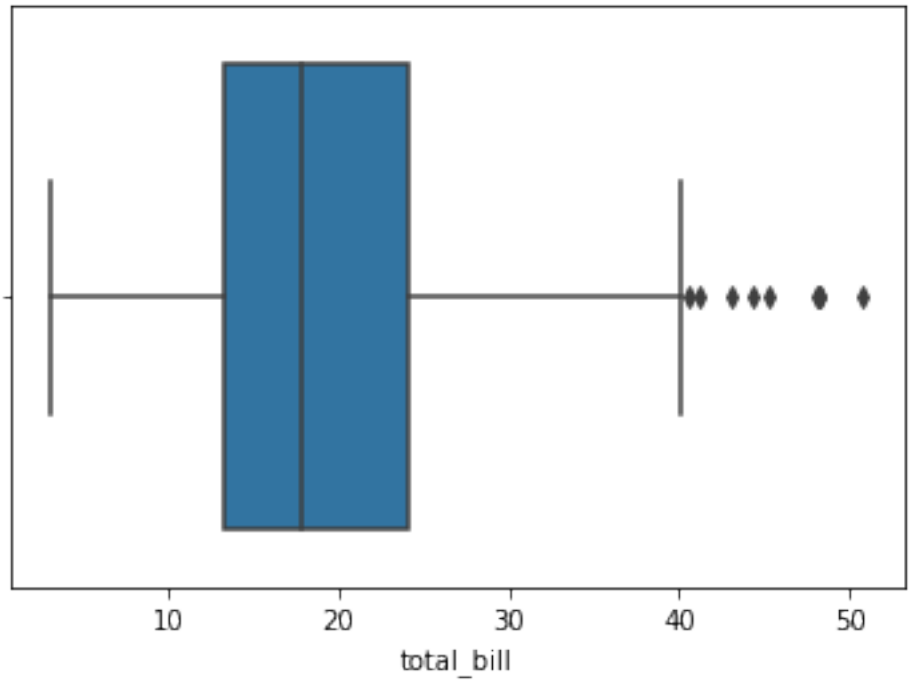
```
            total_bill       tip      size
total_bill    1.000000  0.675734  0.598315
tip           0.675734  1.000000  0.489299
size          0.598315  0.489299  1.000000
```

[12]: <AxesSubplot:>

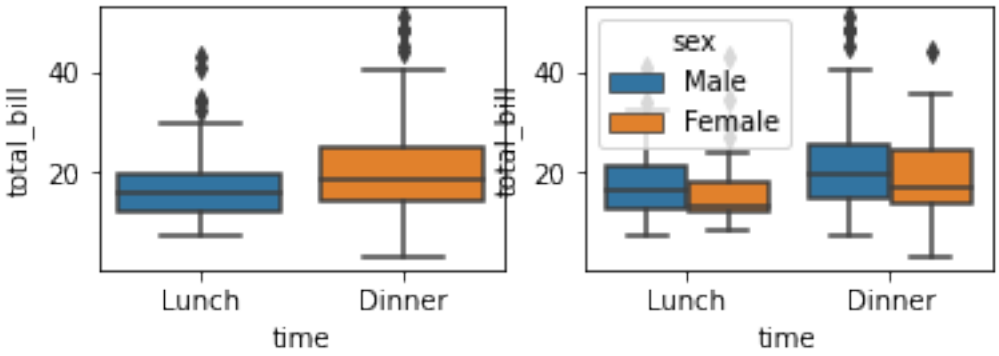


[13]:
```
#box plot------To find outliers
sns.boxplot(x=data1['total_bill'])
```

[13]: <AxesSubplot:xlabel='total_bill'>

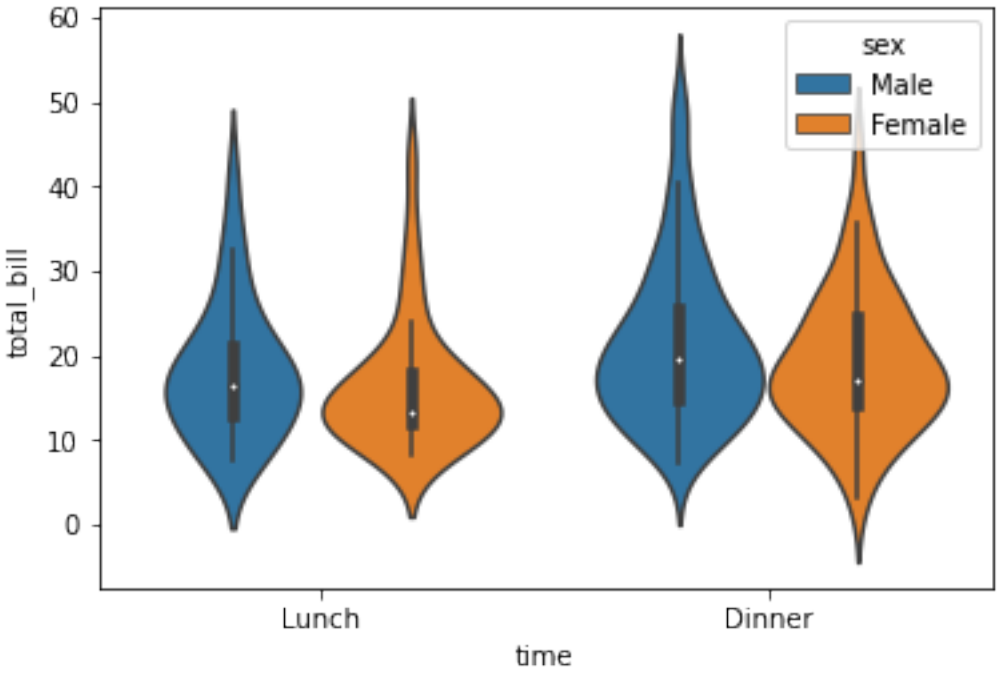


[15]:
```
#group plot
plt.subplot(2,2,1)
sns.boxplot(x=data1['time'],y=data1['total_bill'])
plt.subplot(2,2,2)
sns.boxplot(x=data1['time'],y=data1['total_bill'],hue=data1['sex'])
plt.show()
```



[17]:
```
#Violin plot: same as box plot
sns.violinplot(x=data1['time'],y=data1['total_bill'],hue=data1['sex'])
```

[17]:
```
<AxesSubplot:xlabel='time', ylabel='total_bill'>
```



[24]:
```
#distribution curve
#measure of central tendency
a=pd.Series([2,3,4,5,6,6,12])
print(a.mean())
print(a.median())
print(a.std())
print(a.var())
a_outlier=pd.Series([2,3,4,5,6,6,1000])   #Mean,std deviation have more impact␣
 ↪due to outliers
print(a_outlier.mean())
print(a_outlier.median())
print(a_outlier.std())
print(a_outlier.var())
```

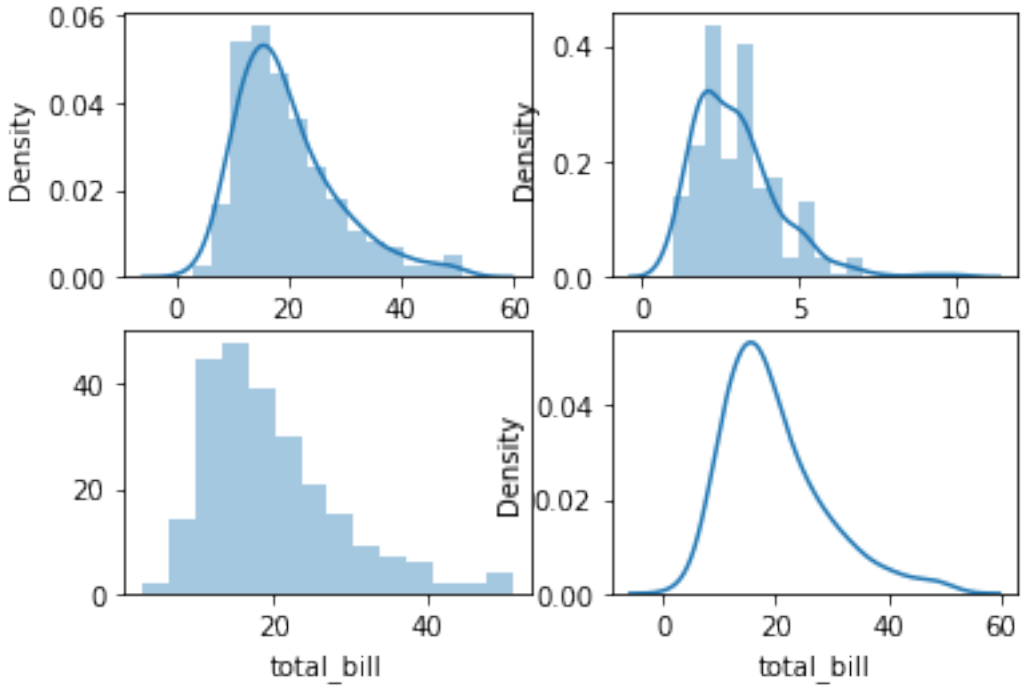
```
5.428571428571429
5.0
3.25868802112869
10.619047619047619
146.57142857142858
5.0
376.3295794658618
141623.9523809524
```

[29]:
```
#distribution curve --- gives both histogram and distribution curve
plt.subplot(2,2,1)
sns.distplot(data1['total_bill'])
plt.subplot(2,2,2)
sns.distplot(data1['tip'])
#if you dont want histogram curve give kde is false
plt.subplot(2,2,3)
sns.distplot(data1['total_bill'],kde=False)
#if you dont want distribution
plt.subplot(2,2,4)
sns.distplot(data1['total_bill'],hist=False)
print(data1.skew())   #-----to get skew value
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619:
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)
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619:
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)
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619:
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)
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619:
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 `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
```

```
total_bill    1.133213
tip           1.465451
size          1.447882
dtype: float64
```



[30]:
```
datanew=pd.read_csv('k_circle_sales (1).csv')
datanew
```

[30]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	\
0	FDA15	9.300	Low Fat	0.016047	
1	DRC01	5.920	Regular	0.019278	
2	FDN15	17.500	Low Fat	0.016760	
3	FDX07	19.200	Regular	0.000000	
4	NCD19	8.930	Low Fat	0.000000	
…	…	…	…	…	
8518	FDF22	6.865	Low Fat	0.056783	
8519	FDS36	8.380	Regular	0.046982	
8520	NCJ29	10.600	Low Fat	0.035186	
8521	FDN46	7.210	Regular	0.145221	
8522	DRG01	14.800	Low Fat	0.044878	

	Item_Type	Item_MRP	Outlet_Identifier	\
0	Dairy	249.8	OUT049	
1	Soft Drinks	48.3	OUT018	
2	Meat	141.6	OUT049	
3	Fruits and Vegetables	182.1	OUT010	
4	Household	53.9	OUT013	
…	…	…	…	
8518	Snack Foods	214.5	OUT013	
8519	Baking Goods	108.2	OUT045	
8520	Health and Hygiene	85.1	OUT035	
8521	Snack Foods	103.1	OUT018	
8522	Soft Drinks	75.5	OUT046	

	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	\
0	1999	Medium	Tier 2	
1	2009	Medium	Tier 2	
2	1999	Medium	Tier 2	
3	1998	NaN	NaN	
4	1987	High	Tier 3	
…	…	…	…	
8518	1987	High	Tier 3	
8519	2002	NaN	NaN	
8520	2004	Small	Tier1	
8521	2009	Medium	Tier 2	
8522	1997	Small	Tier1	

	Outlet_Type	Item_Outlet_Sales	Profit
0	Supermarket Type1	3735.1380	11.5
1	Supermarket Type2	443.4228	14.3
2	Supermarket Type1	2097.2700	14.5
3	Grocery Store	732.3800	13.6
4	Supermarket Type1	994.7052	14.1
…	…	…	…
8518	Supermarket Type1	2778.3834	14.1

```
8519  Supermarket Type1          549.2850    14.2
8520  Supermarket Type1         1193.1136     9.5
8521  Supermarket Type2         1845.5976    14.2
8522  Supermarket Type1          765.6700    14.6

[8523 rows x 13 columns]
```

[31]:
```
datanew.skew()
```

[31]:
```
Item_Weight                  0.097208
Item_Visibility              1.167091
Item_MRP                     0.127390
Outlet_Establishment_Year   -0.396641
Item_Outlet_Sales            1.177531
Profit                      -3.379808
dtype: float64
```

[32]:
```
sns.distplot(datanew['Profit'])
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619:
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)
```

[32]:
```
<AxesSubplot:xlabel='Profit', ylabel='Density'>
```