Seaborn

Data Analytics Project

- Step -1: Business Problem Understanding
  - Understand the Business Problem (Project)
  - Understand Client Requirements (understand they want)
  - Understand what they are expecting for you

Note: If step 1 is not clear never jump into step-2

- Step 2: Data understanding

  - collect data from source
  - Data Exploration
    - understand every column name very clearly (either by research, by asking seniors)
    - understand each variable clearly by applying descriptive statistics
    - observe the complete given data, by applying pandas and seaborn
- Step 3: Data preprocessing or Data prepration

  - Data Cleaning
    - wrong data
    - wrong data type
    - duplicates
    - missing values
    - outliers

After the data cleaning completed , store data as a cleaned data

- Step 4: Analysis
  - Applying various logics as per project requirements
  - infrences/observation in your logic
  - if that observation is important,write in the notes
  - if observation is not important , don't write in notes

List of observations, what you have written in the notes ---> Report

- Step 5:-
- presentation

Step - 1 : Business Problem

- Restaurant owner wants detailed report on sales
- Whatever the data I have provided, From that do analysis and subbmit your report/infrences.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import seaborn as sns

import warnings
warnings.simplefilter("ignore")
```

Step-2.1 : Load the data

In [2]:
```
df = pd.read_csv(r"C:\D-drive\Datascience notes\Notes\05. Data Visualization\tips.c
df
```

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
...	...	...	...	...	...	...	...
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 × 7 columns

Step- 2.2: Data Understanding

- We understand the each and every column name vary clearly (do research)

- Understand the dataset by applying info(),shape,dtypes,columns
- list the continuous, discrete categorical, discrete count
- Observe the data

In [3]:
```
df.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    object 
 3   smoker      244 non-null    object 
 4   day         244 non-null    object 
 5   time        244 non-null    object 
 6   size        244 non-null    int64  
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

In [4]:
```
df['total_bill'].describe()
```

Out[4]:
```
count    244.000000
mean      19.785943
std        8.902412
min        3.070000
25%       13.347500
50%       17.795000
75%       24.127500
max       50.810000
Name: total_bill, dtype: float64
```

In [5]:
```
df['tip'].describe()
```

Out[5]:
```
count    244.000000
mean       2.998279
std        1.383638
min        1.000000
25%        2.000000
50%        2.900000
75%        3.562500
max       10.000000
Name: tip, dtype: float64
```

In [6]:
```
df['sex'].unique()
```

Out[6]:
```
array(['Female', 'Male'], dtype=object)
```

In [7]:
```
df['sex'].value_counts()
```

Out[7]:
```
sex
Male      157
Female     87
Name: count, dtype: int64
```

In [8]:
```
df['smoker'].unique()
```
Out[8]:
```
array(['No', 'Yes'], dtype=object)
```

In [9]:
```
df['smoker'].value_counts()
```
Out[9]:
```
smoker
No     151
Yes     93
Name: count, dtype: int64
```

In [10]:
```
df['day'].unique()
```
Out[10]:
```
array(['Sun', 'Sat', 'Thur', 'Fri'], dtype=object)
```

In [11]:
```
df['day'].value_counts()
```
Out[11]:
```
day
Sat     87
Sun     76
Thur    62
Fri     19
Name: count, dtype: int64
```

In [12]:
```
df['time'].unique()
```
Out[12]:
```
array(['Dinner', 'Lunch'], dtype=object)
```

In [13]:
```
df['time'].value_counts()
```
Out[13]:
```
time
Dinner    176
Lunch      68
Name: count, dtype: int64
```

In [14]:
```
df['size'].unique()
```
Out[14]:
```
array([2, 3, 4, 1, 6, 5], dtype=int64)
```

In [15]:
```
df['size'].value_counts()
```
Out[15]:
```
size
2    156
3     38
4     37
5      5
1      4
6      4
Name: count, dtype: int64
```

In [16]:
```
df.columns
```
Out[16]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='objec
t')
```

In [17]:
```
continuous = ['total_bill','tip']

discrete_categorical = ['sex', 'smoker', 'day', 'time']

discrete_count = ['size']
```

Descriptive Statistics

In [18]:
```
df[continuous].describe()
```

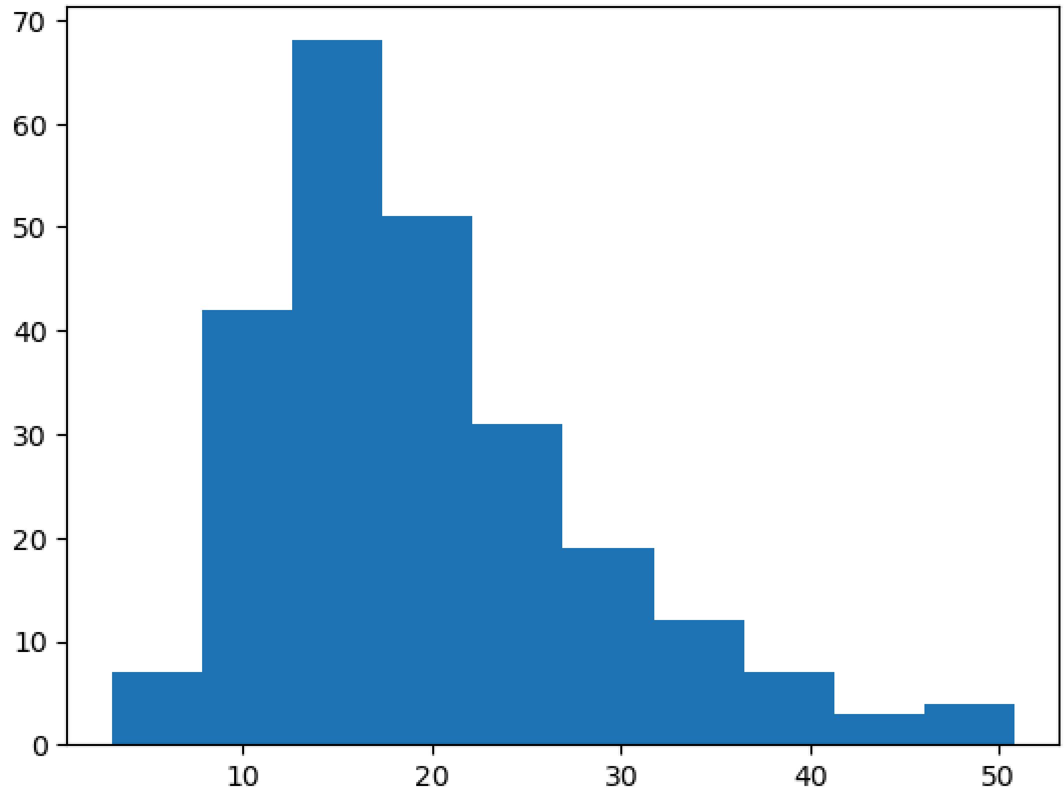
Out[18]:

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

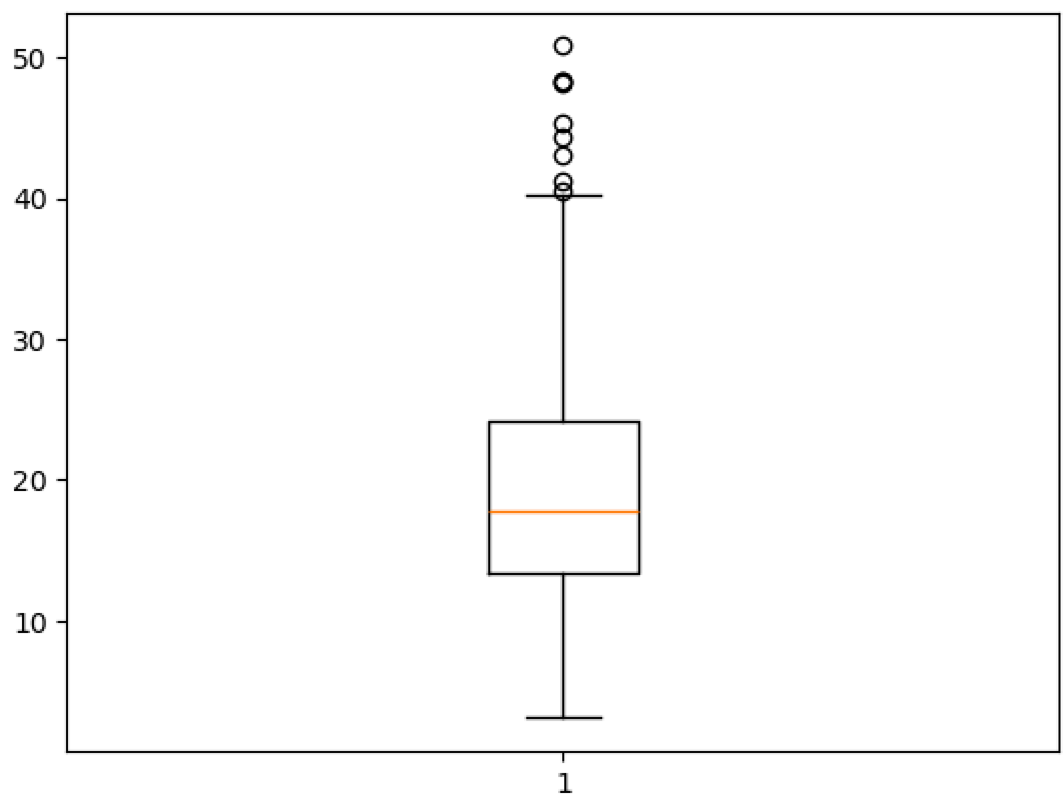
In [19]:
```
df[continuous].skew()
```

Out[19]:
```
total_bill    1.133213
tip           1.465451
dtype: float64
```

In [20]:
```
plt.hist(df['total_bill'],bins = 10)
plt.show()
```



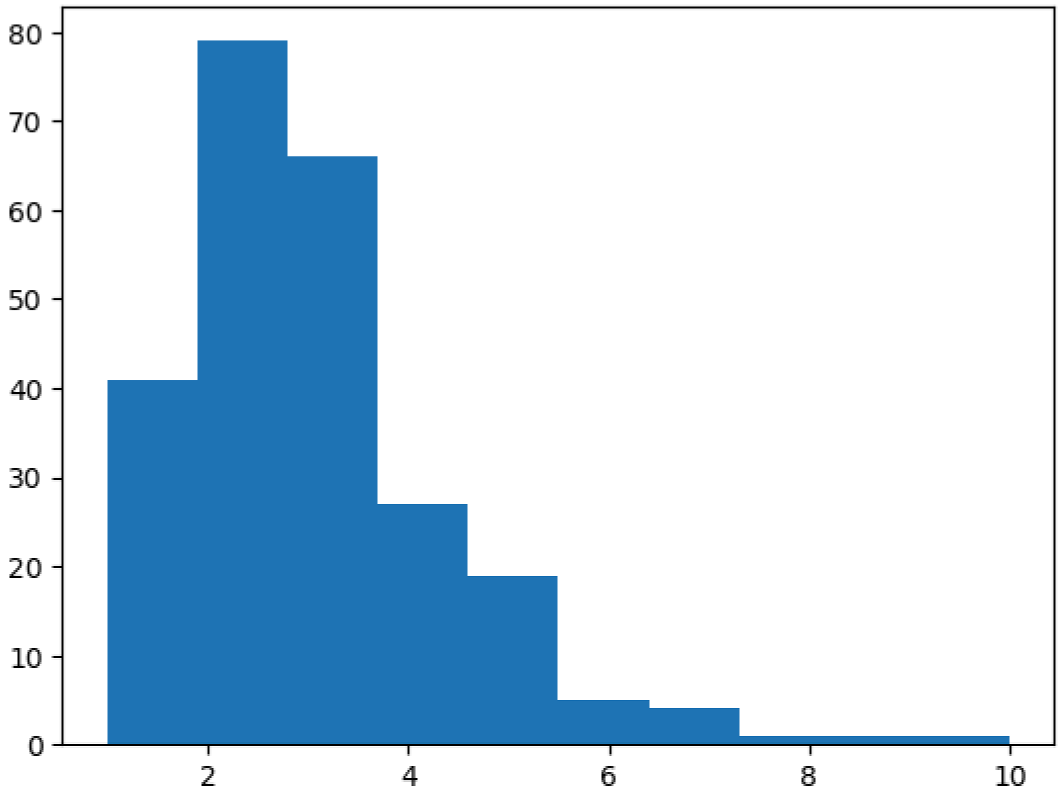
In [21]:
```
plt.boxplot(df['total_bill'])
plt.show()
```



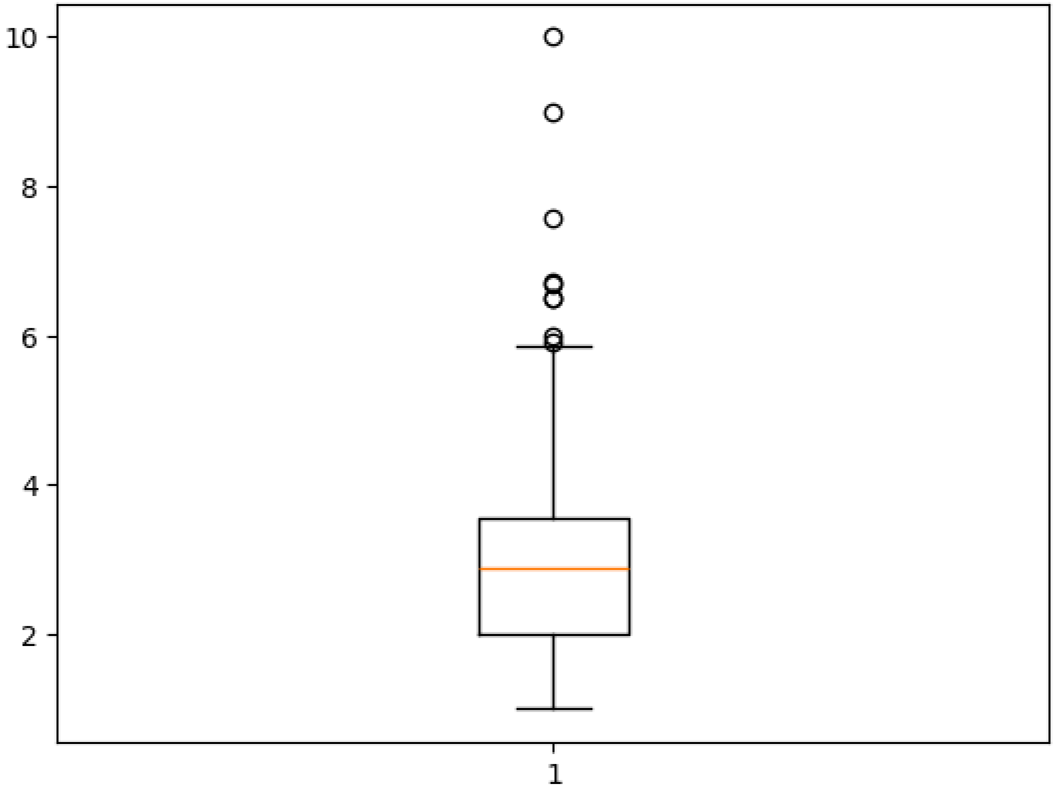
Observations

- total_bill variable is right skewed & also having some outliers
- tip variable is right skewed & also having some outliers

In [22]:
```
plt.hist(df['tip'],bins=10)
plt.show()
```



In [23]:
```
plt.boxplot(df['tip'])
plt.show()
```



In [24]:
```
df[discrete_categorical].describe()
```

Out[24]:

	sex	smoker	day	time
count	244	244	244	244
unique	2	2	4	2
top	Male	No	Sat	Dinner
freq	157	151	87	176

Step - 3 : Data Preprocessing

Drop Duplicates

In [25]:
```
#df = df.drop_duplicates()
```

In [26]:
```
# df.drop_duplicates()   --> there is  chance of repetation , do don't drop duplica
```

Treat Missing values

In [27]:
```
df.isnull().sum()
```

Out[27]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

treat outliers

In [28]:
```
# retrain outliers             why?---> valid (logical) answer
```

Step-4 : Analysis

- Pandas & Seaborn

In [29]:
```
df.groupby('sex')['total_bill'].describe().transpose()
```

Out[29]:

sex	Female	Male
count	87.000000	157.000000
mean	18.056897	20.744076
std	8.009209	9.246469
min	3.070000	7.250000
25%	12.750000	14.000000
50%	16.400000	18.350000
75%	21.520000	24.710000
max	44.300000	50.810000

In [30]:
```
df.groupby('day')['total_bill'].describe().transpose()
```

Out[30]:

day	Fri	Sat	Sun	Thur
count	19.000000	87.000000	76.000000	62.000000
mean	17.151579	20.441379	21.410000	17.682742
std	8.302660	9.480419	8.832122	7.886170
min	5.750000	3.070000	7.250000	7.510000
25%	12.095000	13.905000	14.987500	12.442500
50%	15.380000	18.240000	19.630000	16.200000
75%	21.750000	24.740000	25.597500	20.155000
max	40.170000	50.810000	48.170000	43.110000

In [31]:
```
df.groupby('time')['total_bill'].describe().transpose()
```

Out[31]:

time	Dinner	Lunch
count	176.000000	68.000000
mean	20.797159	17.168676
std	9.142029	7.713882
min	3.070000	7.510000
25%	14.437500	12.235000
50%	18.390000	15.965000
75%	25.282500	19.532500
max	50.810000	43.110000

In [32]:
```
pd.crosstab(df['sex'],df['time'])
```

Out[32]:

time	Dinner	Lunch
sex		
Female	52	35
Male	124	33

In [33]:
```
pd.crosstab(df['sex'],df['day'])      # apply on two categorical variable only
```

Out[33]:

day	Fri	Sat	Sun	Thur
sex				
Female	9	28	18	32
Male	10	59	58	30

Plot's for continuous Data

1. Univariate (Single Variable)

- Histogram
- Kde plot
- Boxplot

Bivariate(plot between two Variables)

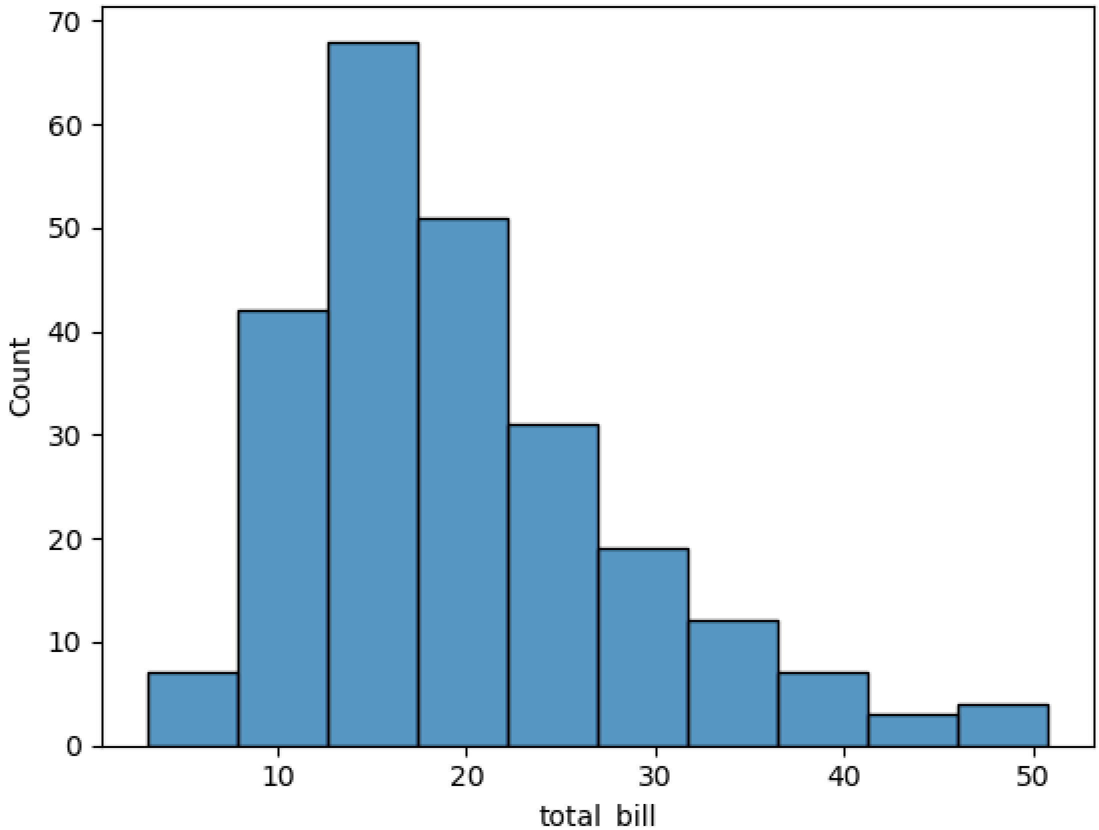
```
- Scatter plot
- Line plot
- Joint plot
- violin plot
```

Multivariate(More then 2 Variables)

```
- Psir Plot
- Heat Plot
```

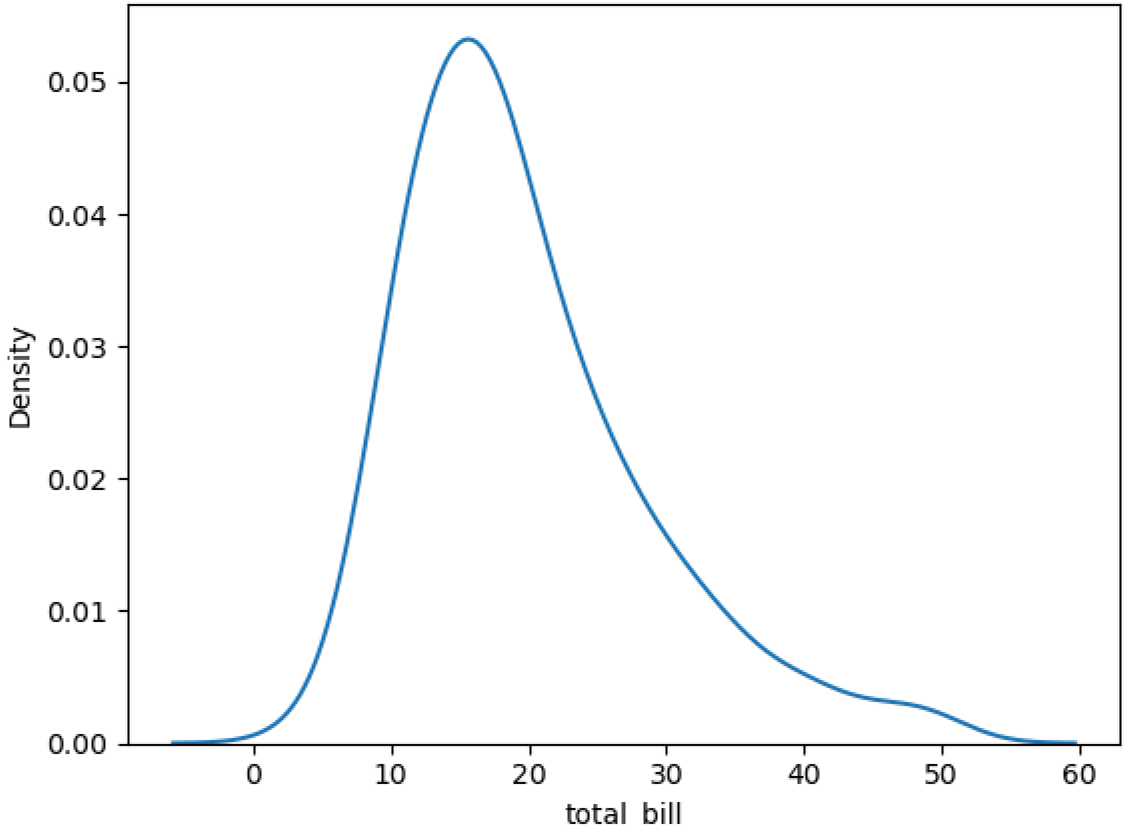
Histogram/Distribution plot

In [34]:
```
sns.histplot(df['total_bill'],bins=10)
plt.show()
```



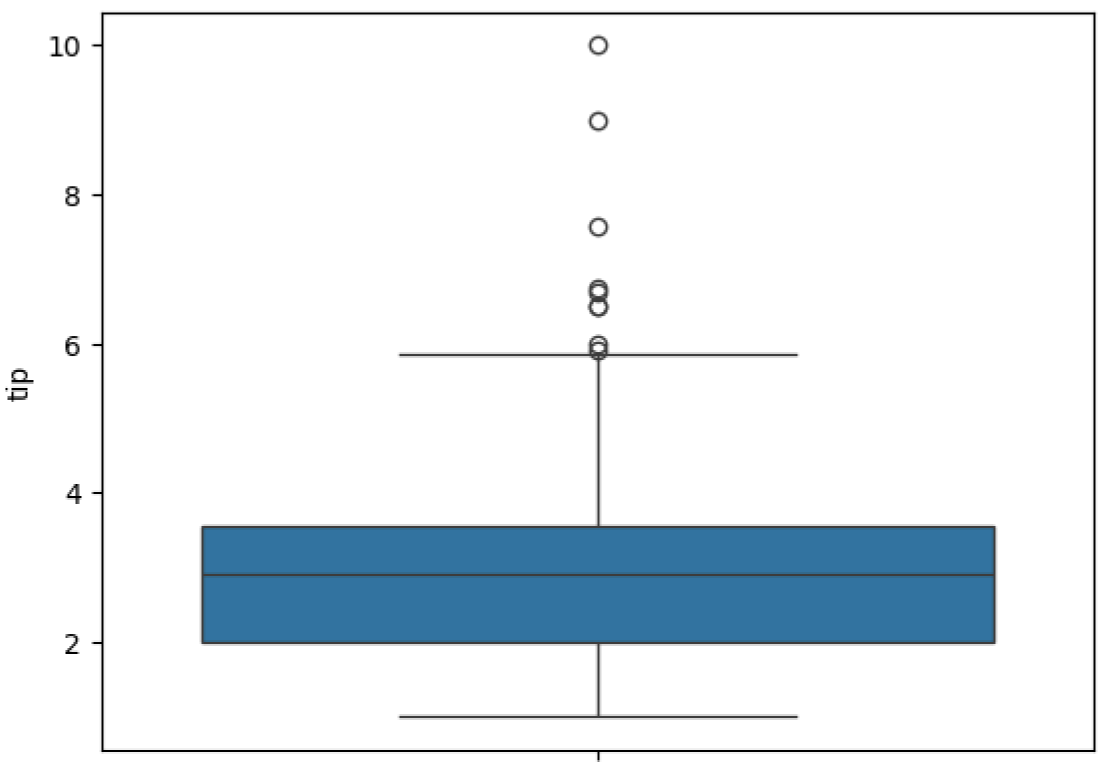
- more no.of customers make bill amount between 10 to 20
- less no.of customers make bill amount > 40$

In [35]:
```
sns.kdeplot(df['total_bill'])
plt.show()
```

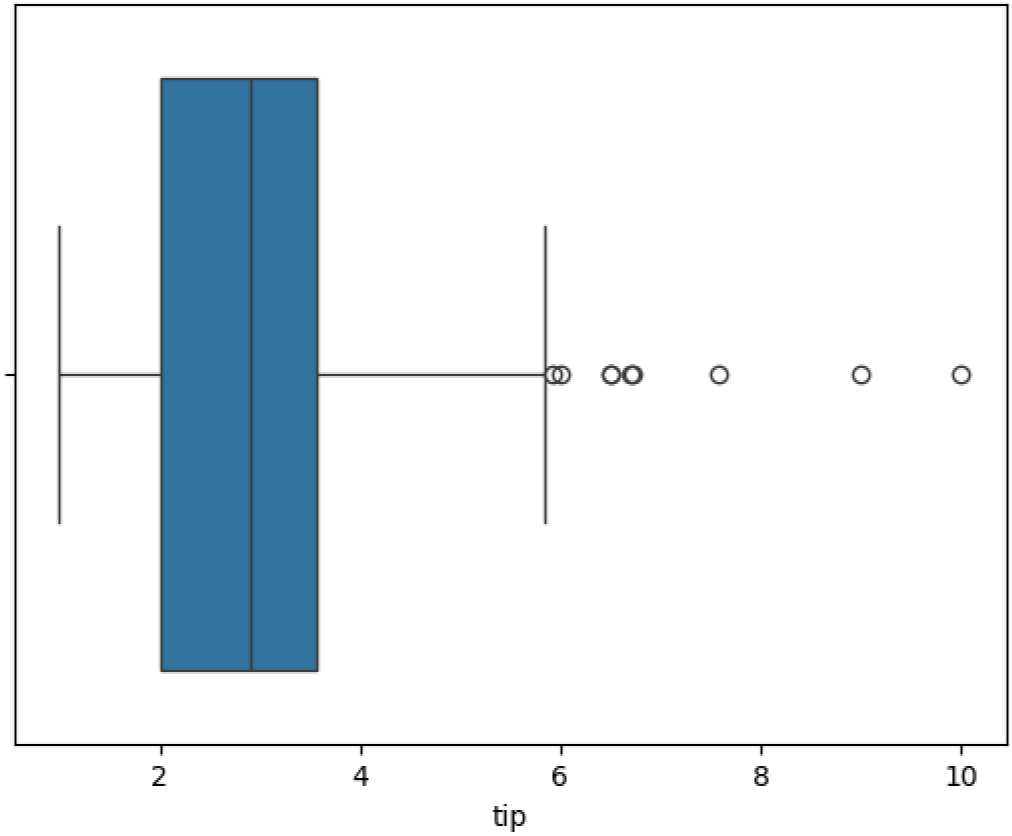


Box plot

In [36]:
```
sns.boxplot(y=df['tip'])
plt.show()
```

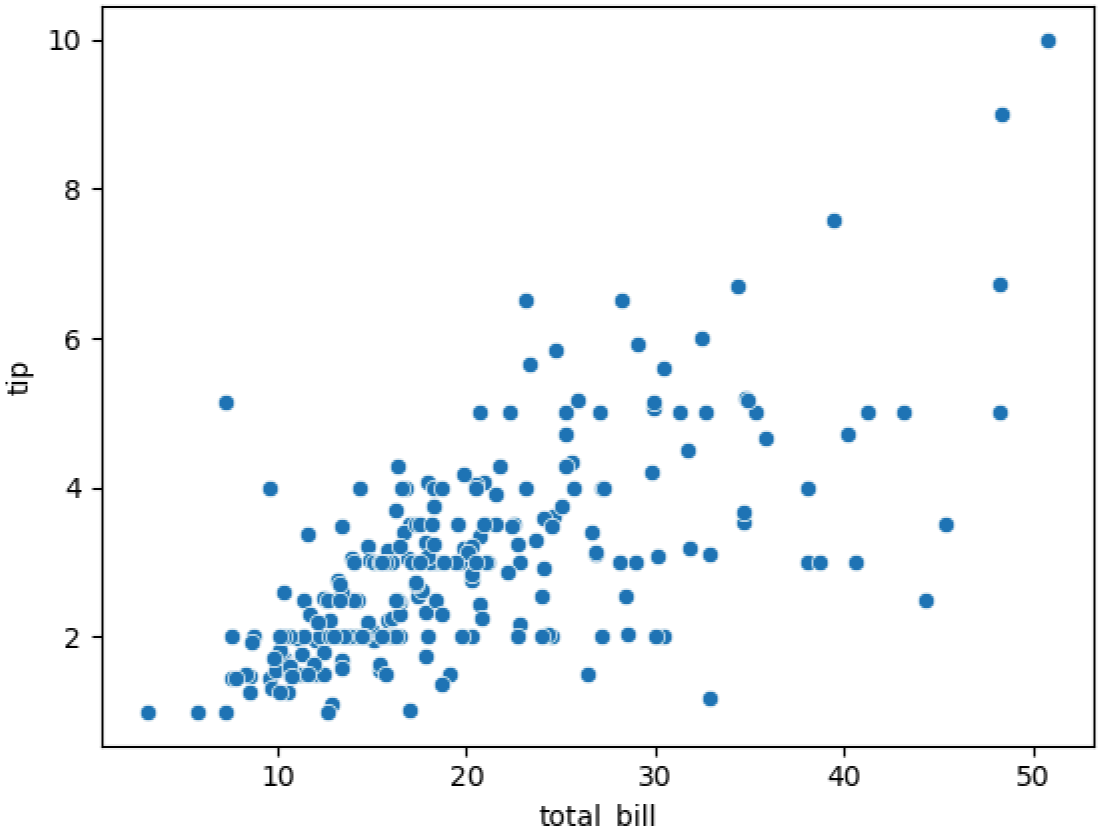


In [37]:
```
sns.boxplot(x=df['tip'])
plt.show()
```



Scatter Plot

In [38]:
```
sns.scatterplot(x=df['total_bill'],y=df['tip'])
plt.show()
```

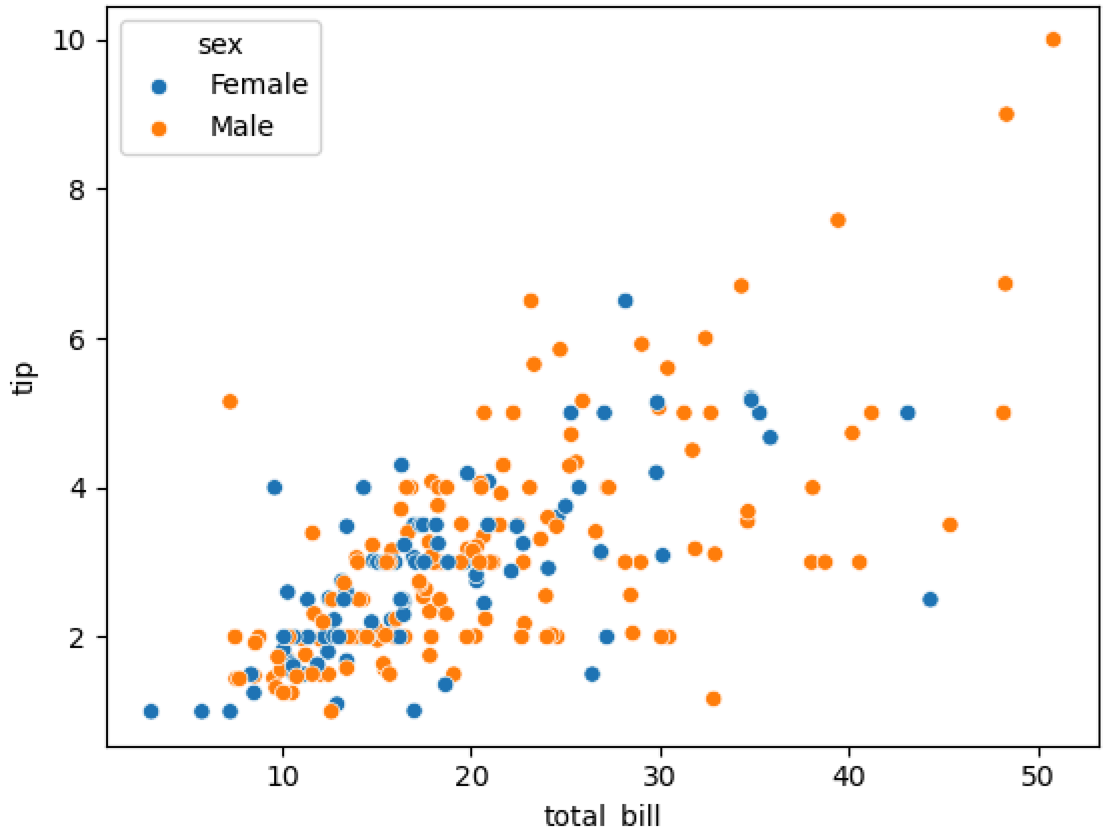


scatter plot using 3 variables

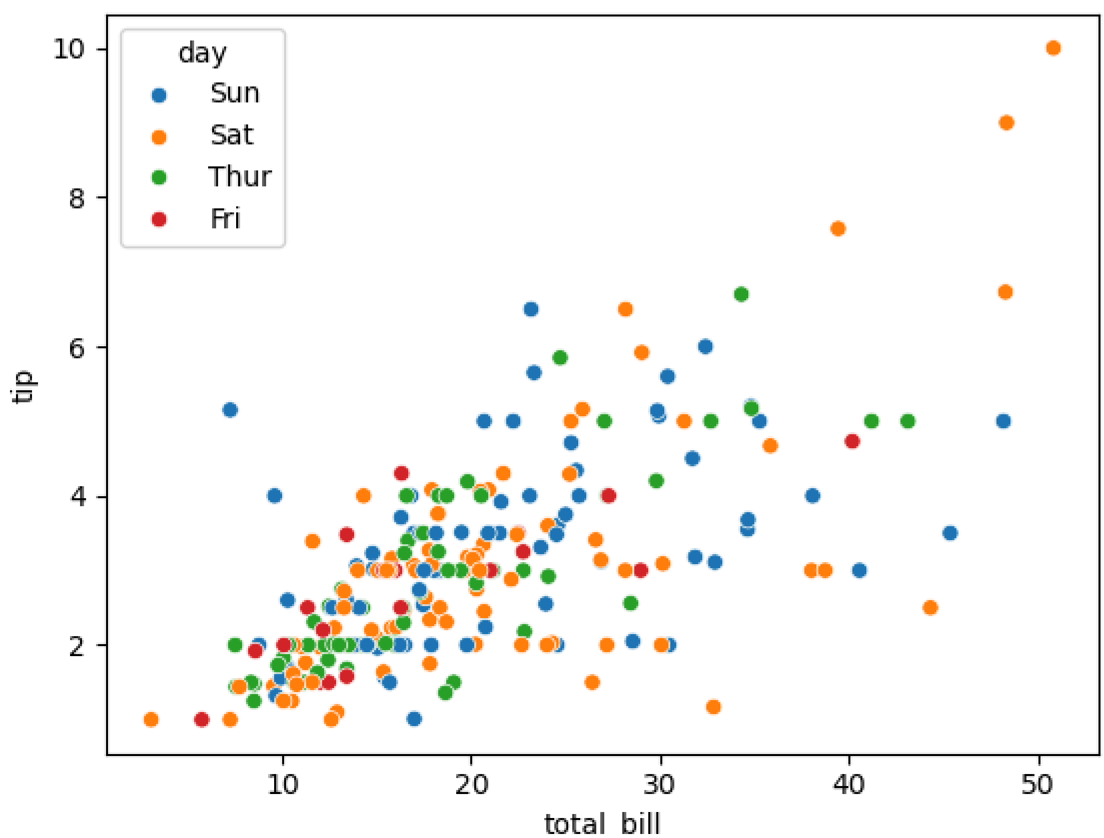
- 2 Continuous Variables
- 1 Discrete Variables

- Basically scatter plot is applied on 2 continuous variables, if you want 3rd variable,it should be compulsorly discrete

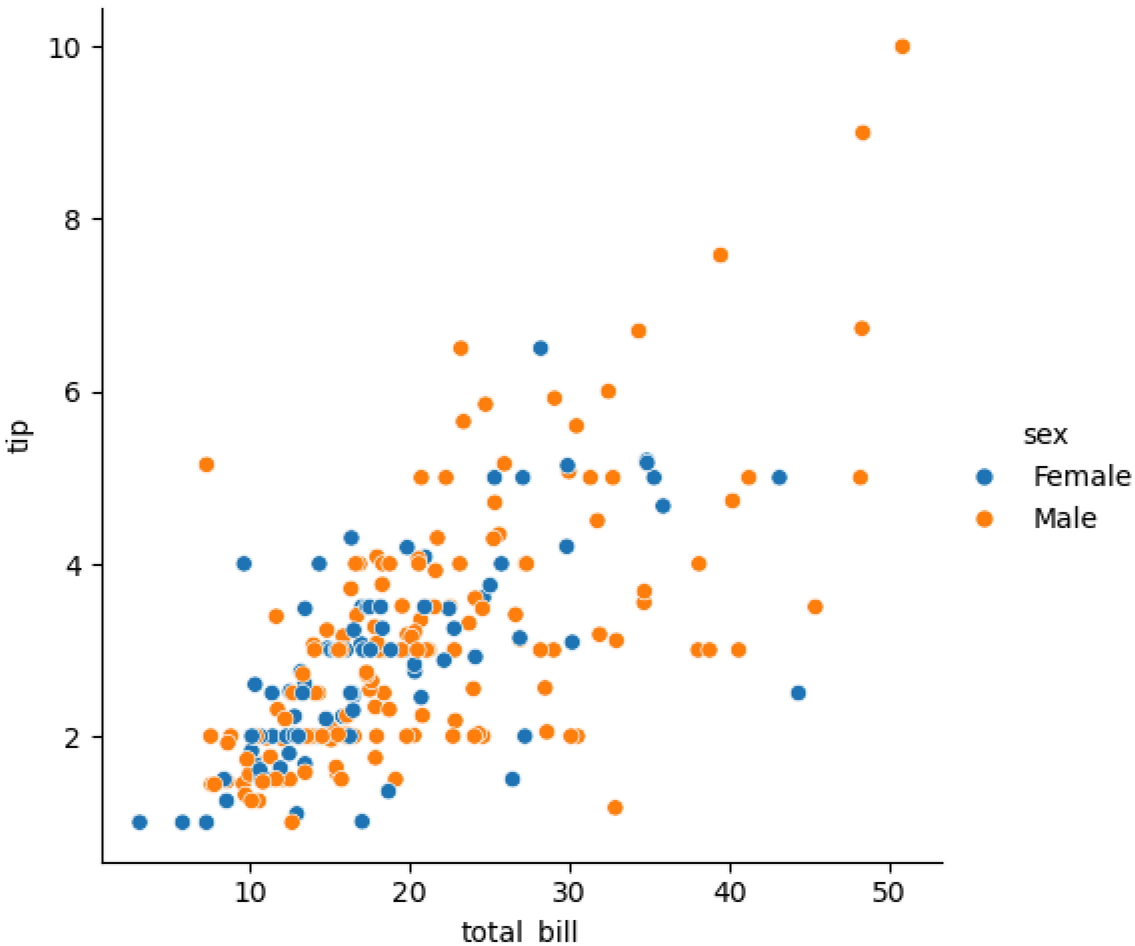
In [39]:
```
sns.scatterplot(x=df['total_bill'],y=df['tip'],hue=df['sex'])
plt.show()
```



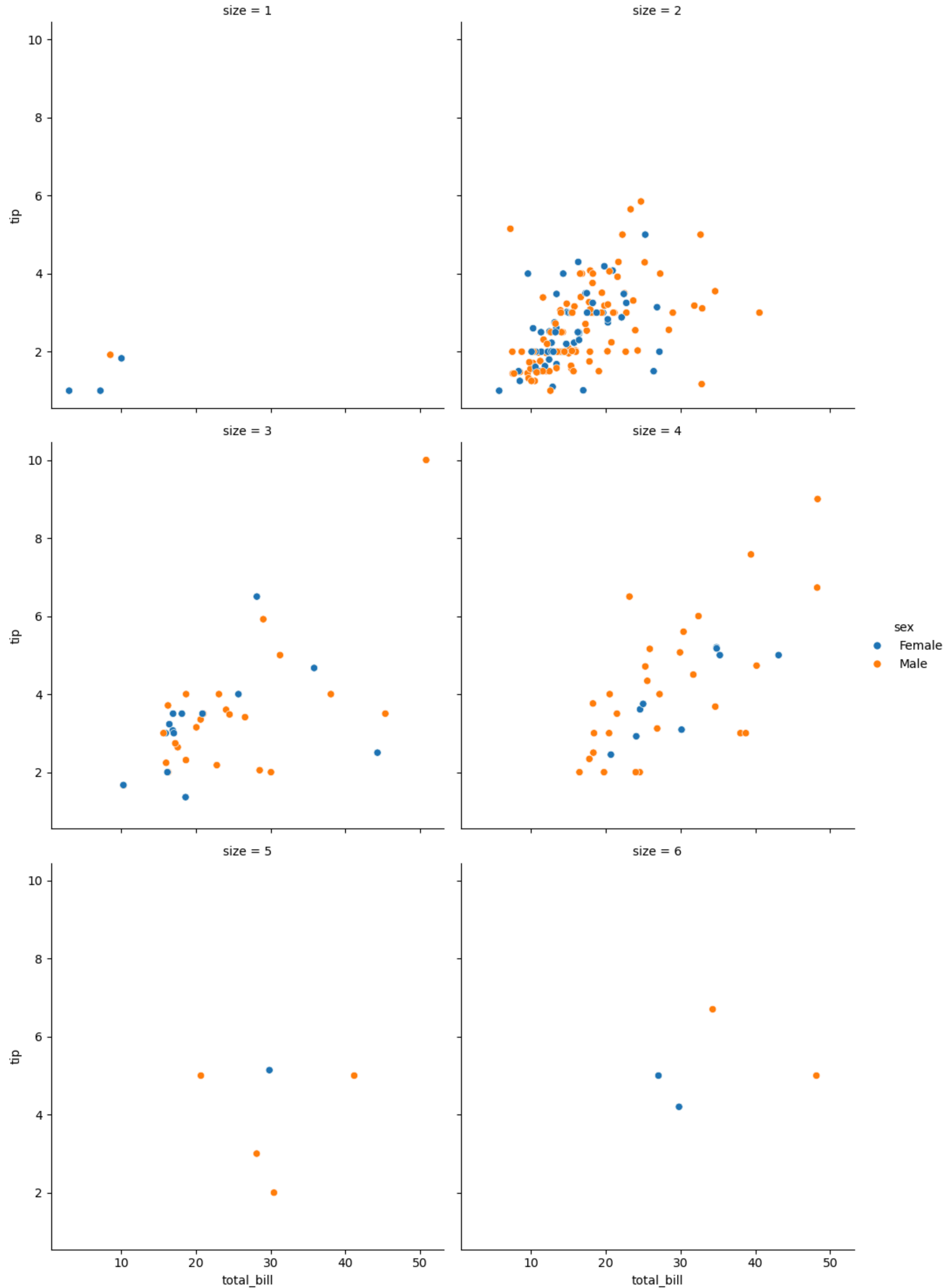
In [40]:
```
sns.scatterplot(x=df['total_bill'],y=df['tip'],hue=df['day'])
plt.show()
```



In [41]:
```
sns.relplot(x=df['total_bill'],y=df['tip'],hue=df['sex'])
plt.show()
```



In [42]:
```
sns.relplot(x=df['total_bill'],y=df['tip'],col=df['size'],col_wrap=2,hue=df['sex'])
plt.show()
```



Line Plot

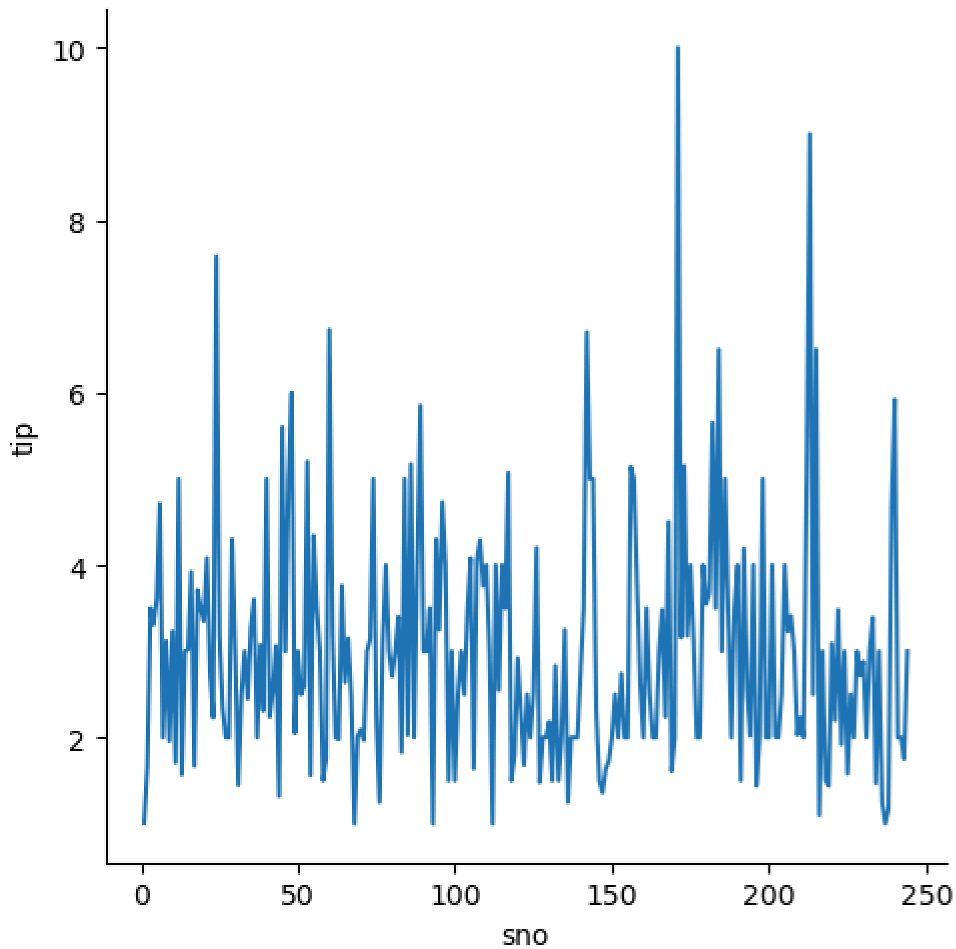
In [43]:
```
df['sno'] = pd.DataFrame(np.arange(1,245))
df
```

Out[43]:

	total_bill	tip	sex	smoker	day	time	size	sno
0	16.99	1.01	Female	No	Sun	Dinner	2	1
1	10.34	1.66	Male	No	Sun	Dinner	3	2
2	21.01	3.50	Male	No	Sun	Dinner	3	3
3	23.68	3.31	Male	No	Sun	Dinner	2	4
4	24.59	3.61	Female	No	Sun	Dinner	4	5
...	...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3	240
240	27.18	2.00	Female	Yes	Sat	Dinner	2	241
241	22.67	2.00	Male	Yes	Sat	Dinner	2	242
242	17.82	1.75	Male	No	Sat	Dinner	2	243
243	18.78	3.00	Female	No	Thur	Dinner	2	244

244 rows × 8 columns

In [44]:
```
sns.relplot(x = 'sno', y = 'tip', data = df, kind='line')
plt.show()
```

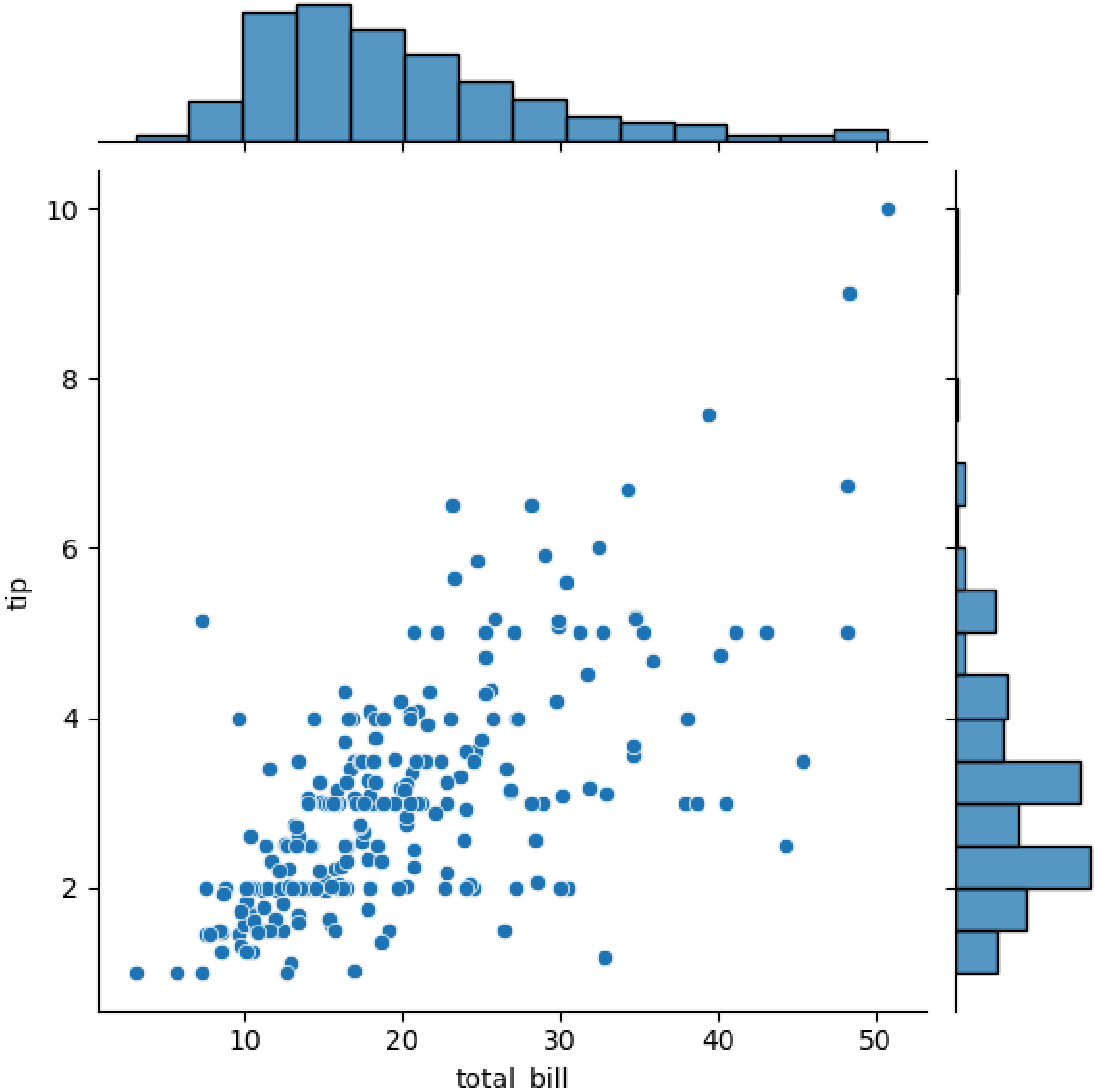


In [45]:
```
df.drop('sno',axis=1,inplace=True)
```

Joint Plot

- A join plot allows to study the relationship between 2 numeric variables.The central chart display there corelation it is usually a scatter plot, a hexbin plot, 2D histogram or a 2D density plot.

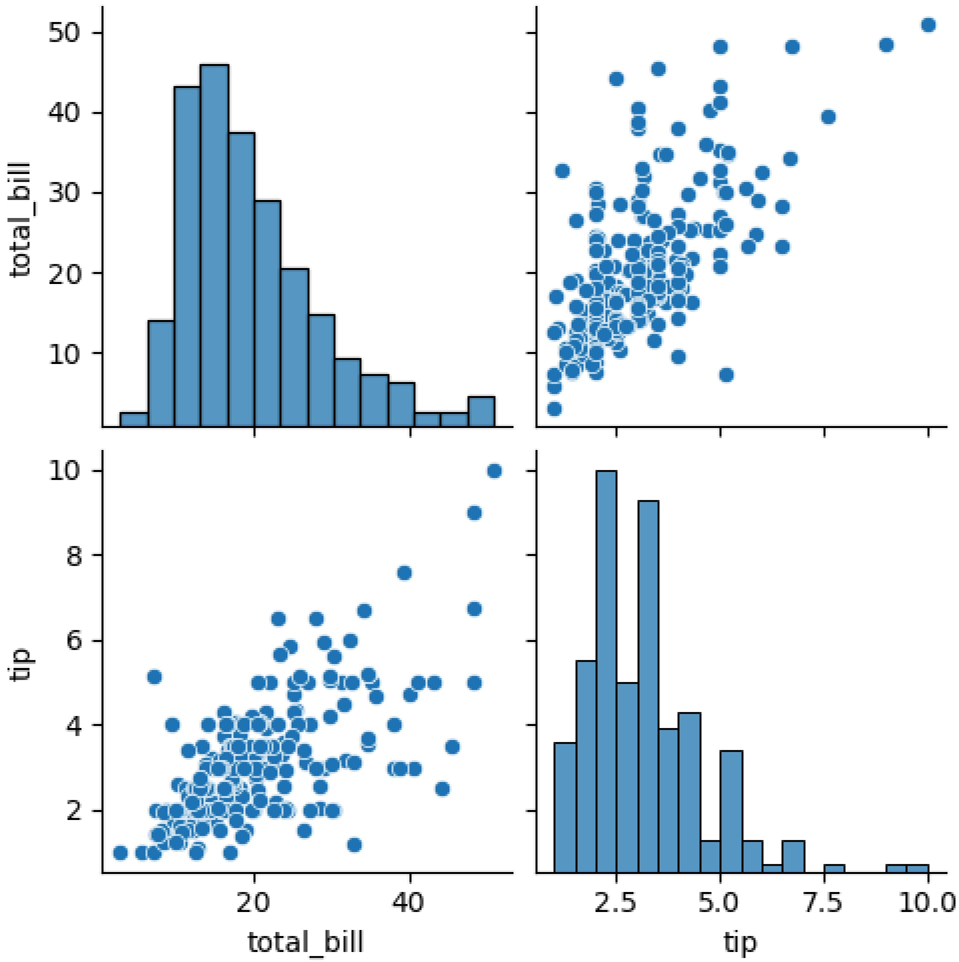
In [46]:
```
sns.jointplot(x='total_bill',y='tip',data = df)
plt.show()
```



Pair plot - Multiple continuous variables

A 'pairs plot' is also known as a scatterplot, in which one variable in the same data row is mached with another variables value, like this , pair plot are just elaborations on this , showing all variables paired with all other variables

In [47]:
```
sns.pairplot(df,vars=continuous)
plt.show()
```



Heat Map

A heatmap uses colored cells to represent relation between variables

Heatmap (for Correlation)

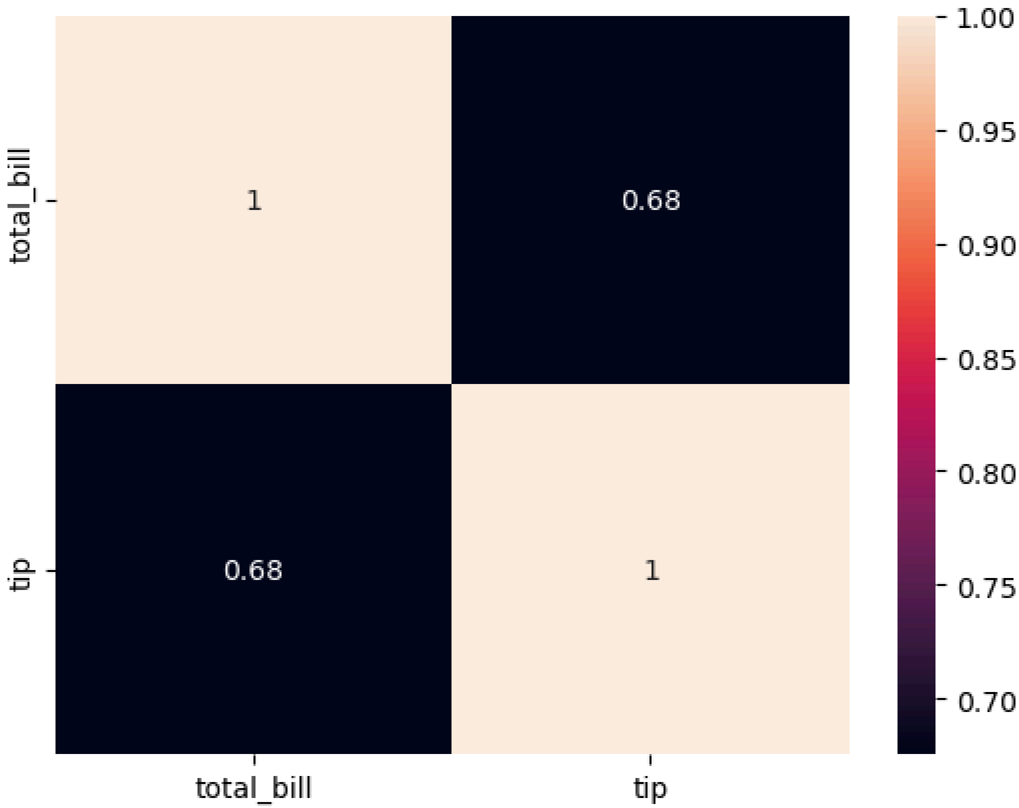
- A corelation heatmap uses colored cells to show a 2D correlation matrix (table) between two neumeric dimensions.
- It is very important in Feture Selection

In [48]:
```
c_m = df[continuous].corr()
c_m
```

Out[48]:

	total_bill	tip
total_bill	1.000000	0.675734
tip	0.675734	1.000000

In [49]:
```
sns.heatmap(c_m,annot=True)
plt.show()
```



Plot's for Discrete

1. Univariate (Single Variable)
   - Pie plot
   - Bar plot
   - Countplot
2. Bivariate (plot between two variables)
   - Boxplot---> One discrete variable & one continuous variable

CountPlot

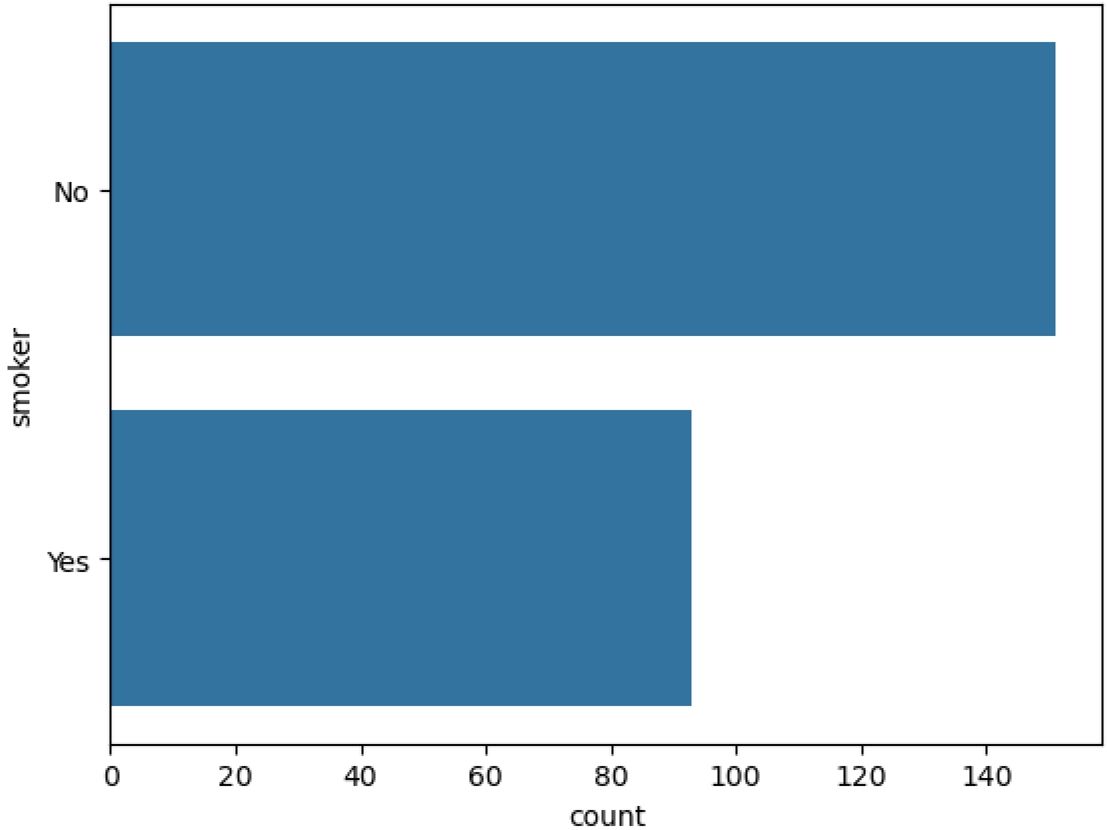
In [50]:
```
df['smoker'].unique()
```

Out[50]:
```
array(['No', 'Yes'], dtype=object)
```

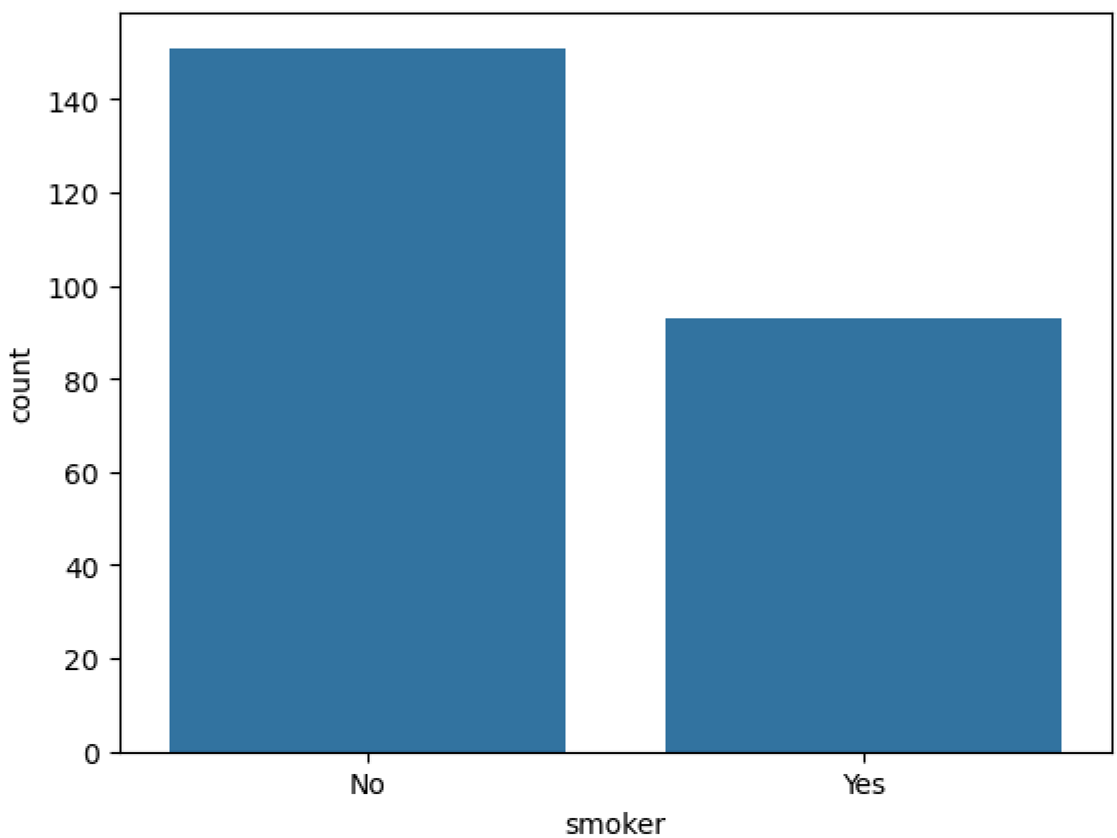
In [51]:
```
df['smoker'].value_counts()
```

Out[51]:
```
smoker
No     151
Yes     93
Name: count, dtype: int64
```

In [52]:
```
sns.countplot(y= 'smoker',data=df)
plt.show()
```



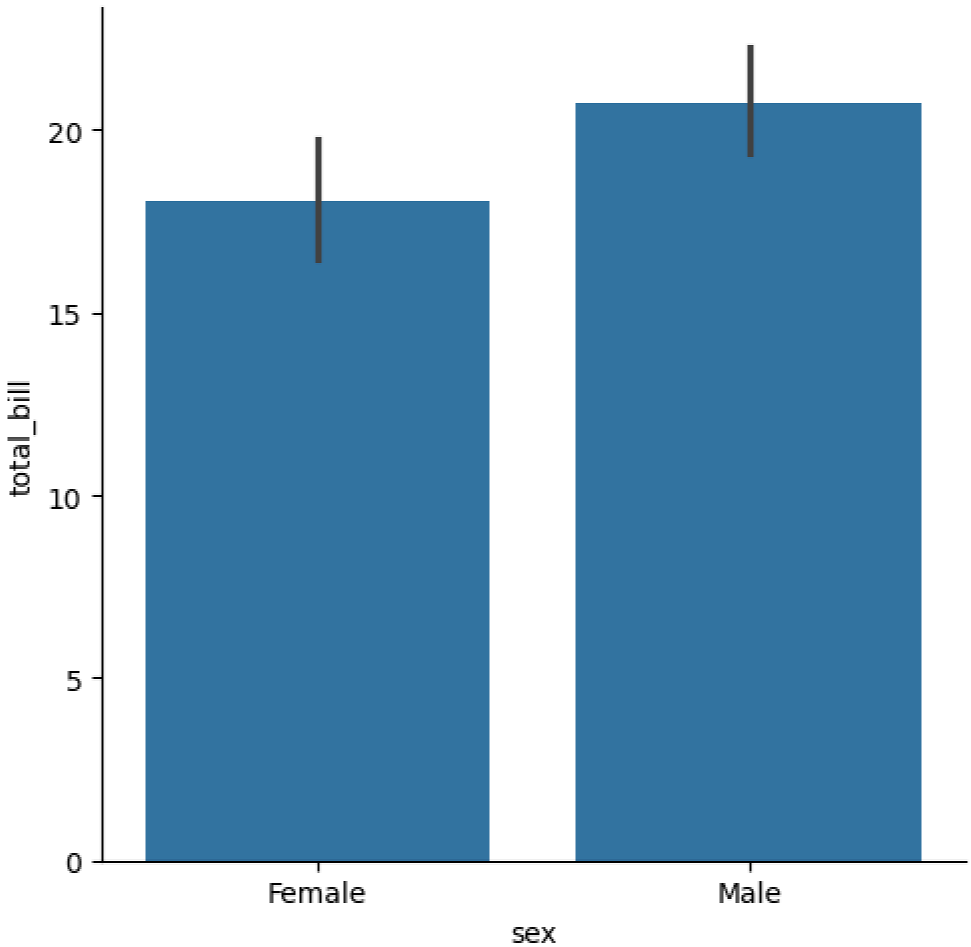
In [53]:
```
sns.countplot(x=df['smoker'])
plt.show()
```



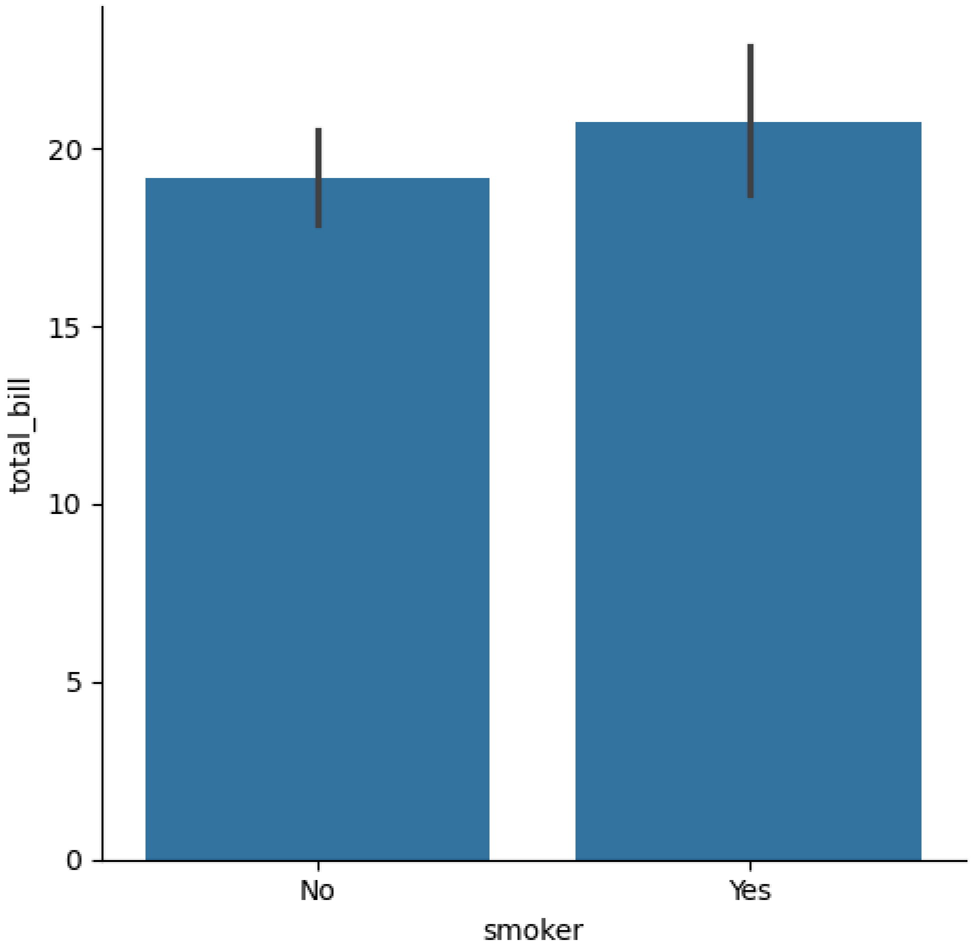
In [54]:
```
df.groupby('sex')['total_bill'].mean()
```

Out[54]:
```
sex
Female    18.056897
Male      20.744076
Name: total_bill, dtype: float64
```

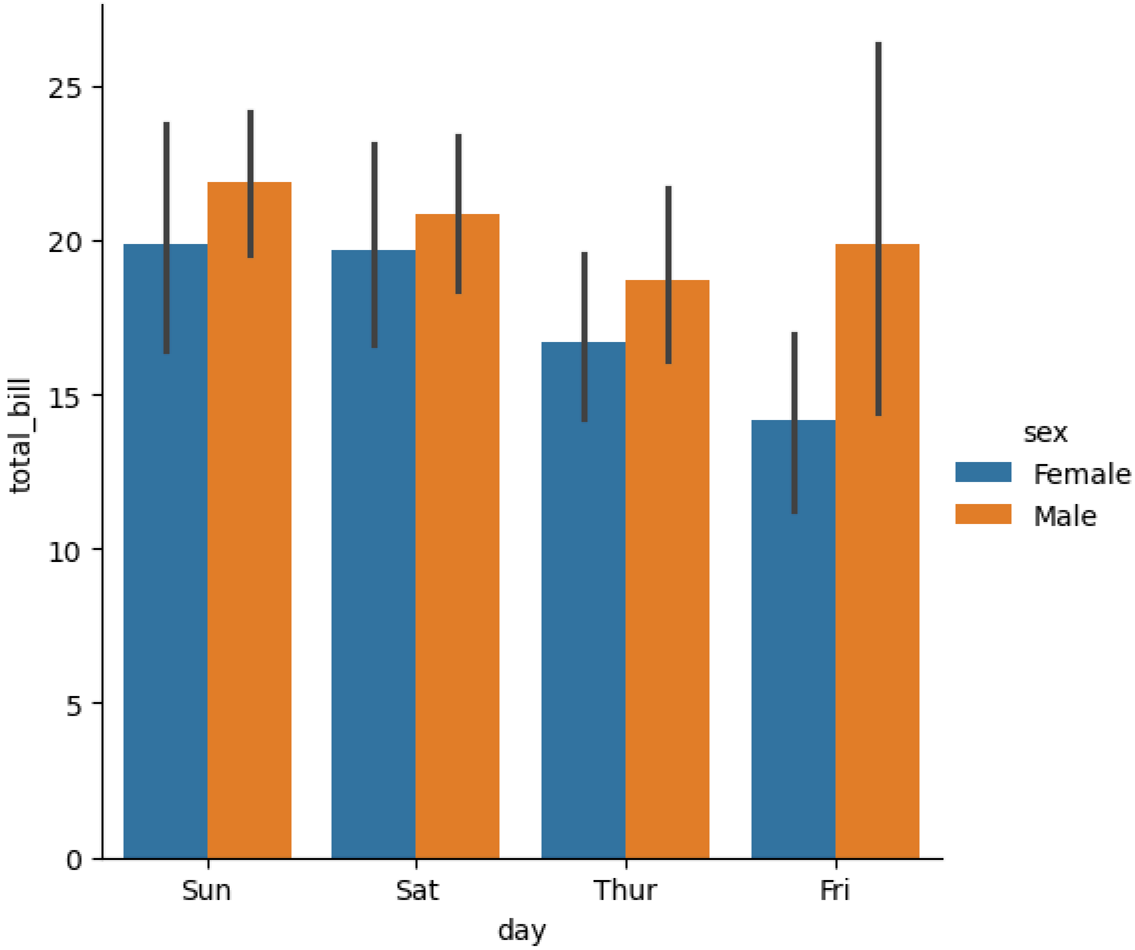
In [55]:
```
sns.catplot(x='sex',y='total_bill',data=df,kind='bar')
plt.show()
```



In [56]:
```
sns.catplot(x ='smoker',y='total_bill', data= df,kind='bar')
plt.show()
```



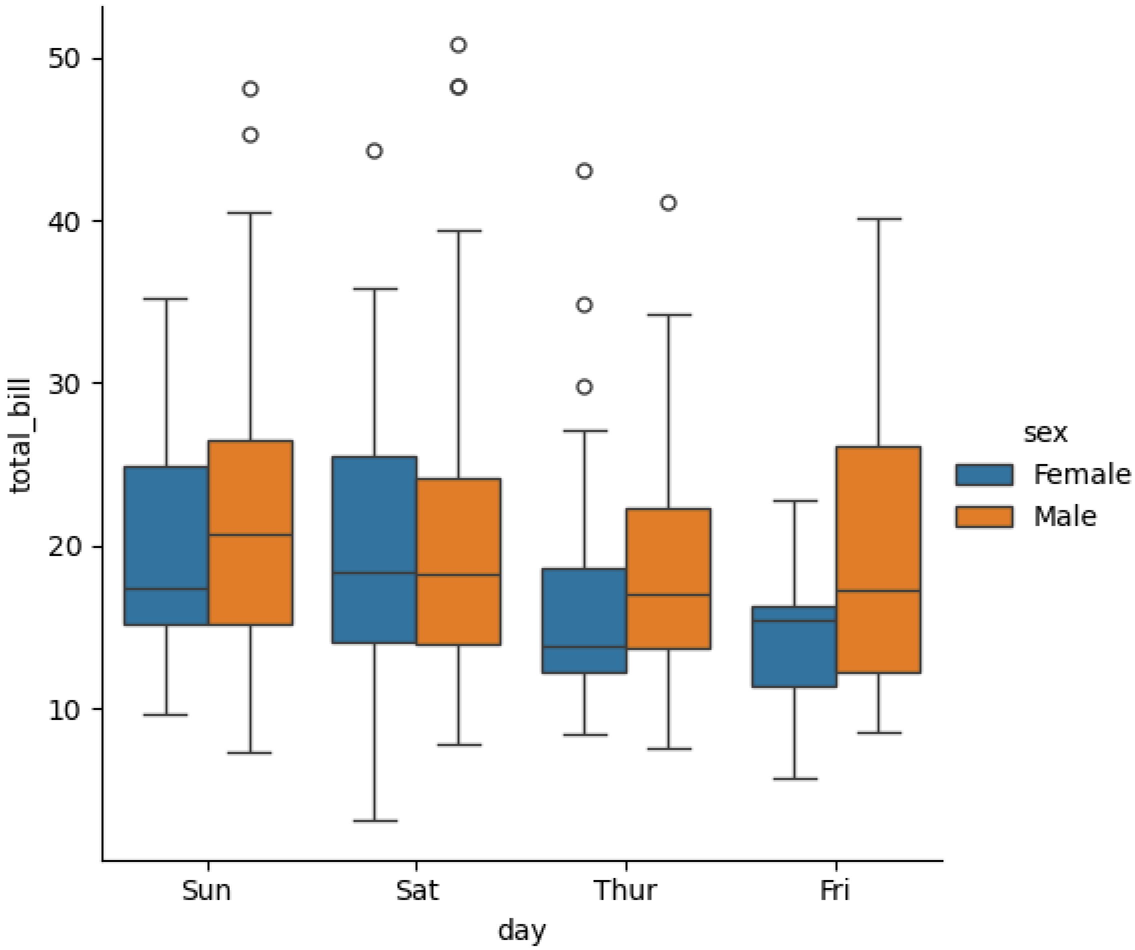
In [57]:
```
sns.catplot(x = 'day' , y = 'total_bill', data = df, kind='bar',hue='sex')
plt.show()
```



In [58]:
```
sns.catplot(x='day', y='total_bill', data = df, kind = 'box', hue ='sex')
plt.show(



)
```



In [ ]: