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

import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
import seaborn as sns
```

In [3]:
```
df = pd.read_csv("tips.csv")
df.head()
```

Out[3]:

	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

Dataset Understanding

In [4]:
```
df.shape
```

Out[4]:
```
(244, 7)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='o
bject')
```

In [6]:
```
df.dtypes
```

Out[6]:
```
total_bill    float64
tip           float64
sex            object
smoker         object
day            object
time           object
size            int64
dtype: object
```

In [7]:
```
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 [8]:
```
df.isnull().sum()
```

Out[8]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

Exploratory Data Analysis

In [9]:
```
df.describe()
```

Out[9]:

	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 [10]:
```
df.describe(include="all")
```

Out[10]:

	total_bill	tip	sex	smoker	day	time	size
count	244.000000	244.000000	244	244	244	244	244.000000
unique	NaN	NaN	2	2	4	2	NaN
top	NaN	NaN	Male	No	Sat	Dinner	NaN
freq	NaN	NaN	157	151	87	176	NaN
mean	19.785943	2.998279	NaN	NaN	NaN	NaN	2.569672
std	8.902412	1.383638	NaN	NaN	NaN	NaN	0.951100
min	3.070000	1.000000	NaN	NaN	NaN	NaN	1.000000
25%	13.347500	2.000000	NaN	NaN	NaN	NaN	2.000000
50%	17.795000	2.900000	NaN	NaN	NaN	NaN	2.000000
75%	24.127500	3.562500	NaN	NaN	NaN	NaN	3.000000
max	50.810000	10.000000	NaN	NaN	NaN	NaN	6.000000

In [11]:
```
df['time'].value_counts()
```

Out[11]:
```
Dinner    176
Lunch      68
Name: time, dtype: int64
```

1. Plot's for Continous Data

1. Univariate (Single Variable)

- histogram or dist()
- boxplot()

2. Bivariate (plot between two Variables)

- scatterplot()
- lineplot()
- jointplot()
- violinplot()

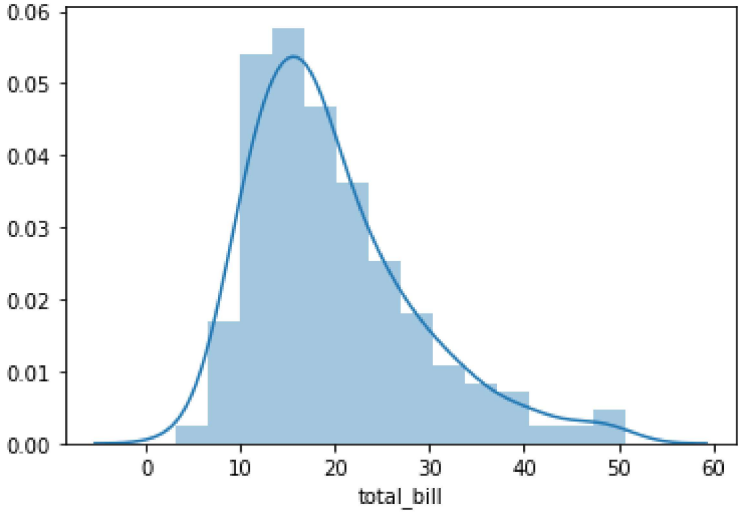
3. Multivariate (More than 2 Variables)

- pairplot()
- heatmap()

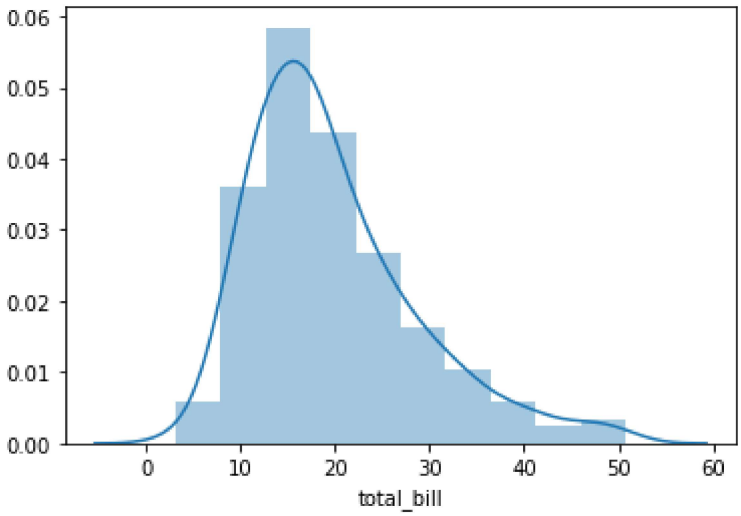
Histogram/Dist plot

Dist plot helps us to check the distribution of a feature

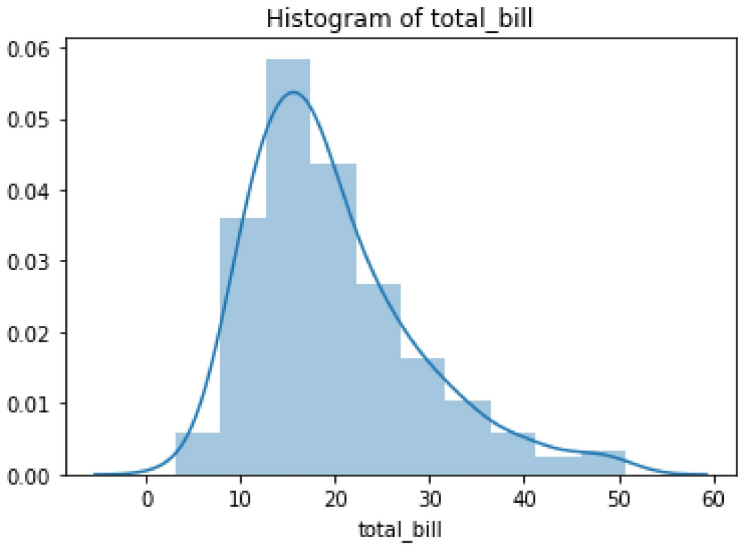
In [12]:
```
sns.distplot(df['total_bill'])
plt.show()
```



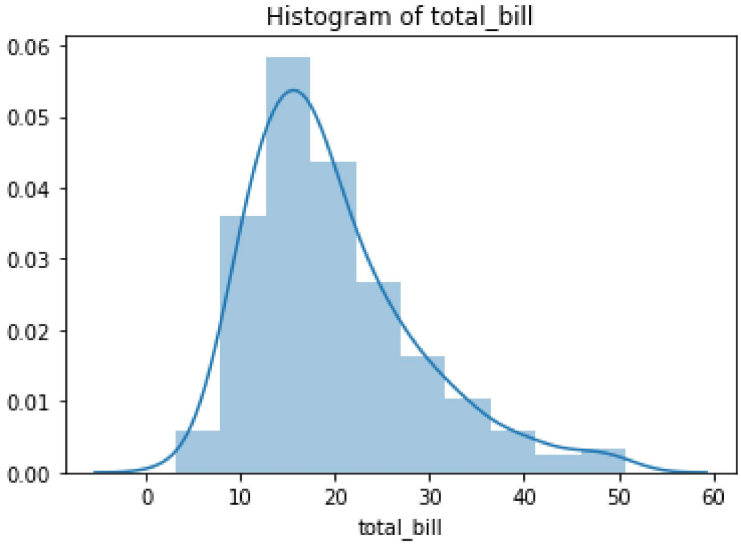
In [13]:
```
sns.distplot(df['total_bill'],bins=10,kde=True)
plt.show()
```



In [14]:
```
sns.distplot(df['total_bill'],bins=10,kde=True).set_title("Histogram of total_
bill")
plt.title("Histogram of total_bill")
plt.show()
```



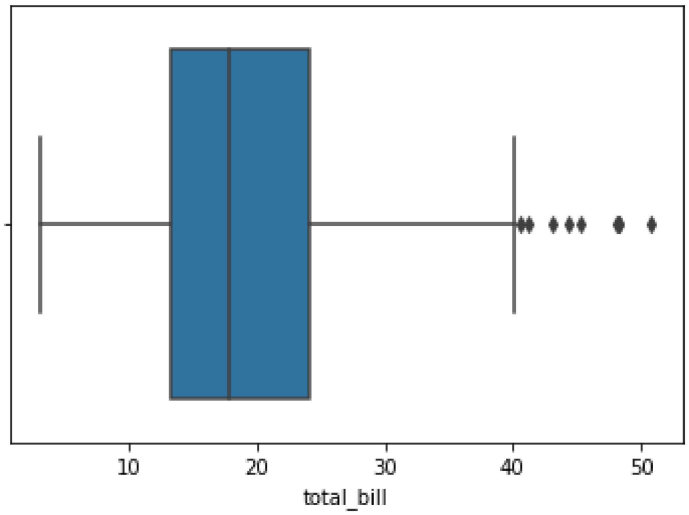
In [15]:
```
sns.distplot(df['total_bill'],bins=10,kde=True)
plt.title("Histogram of total_bill")
plt.show()
```



Box plot

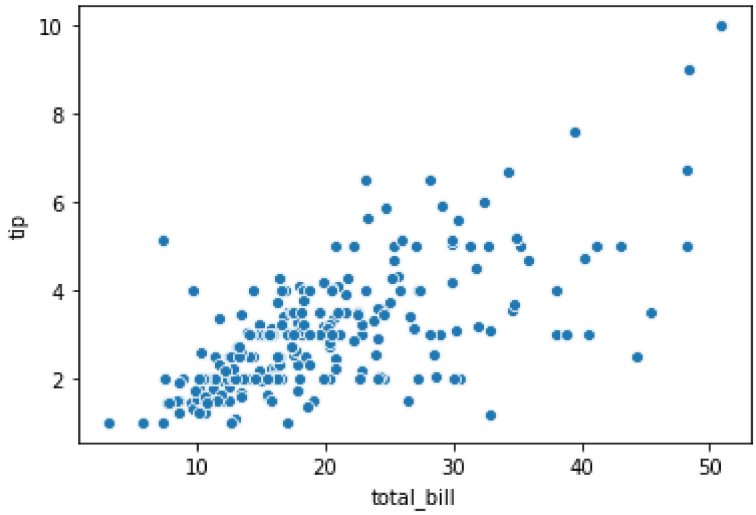
A box and whisker plot (sometimes called a boxplot) is a graph that presents information from a five-number summary.

In [16]:
```
sns.boxplot(df["total_bill"])
plt.show()
```

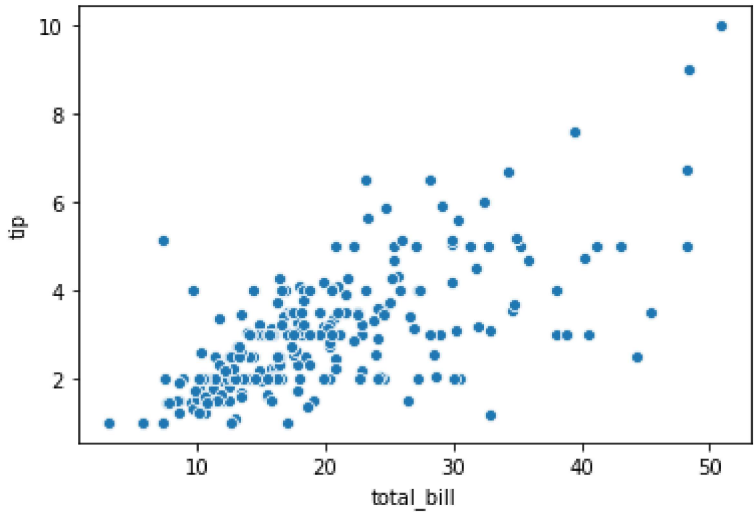


Scatter Plot

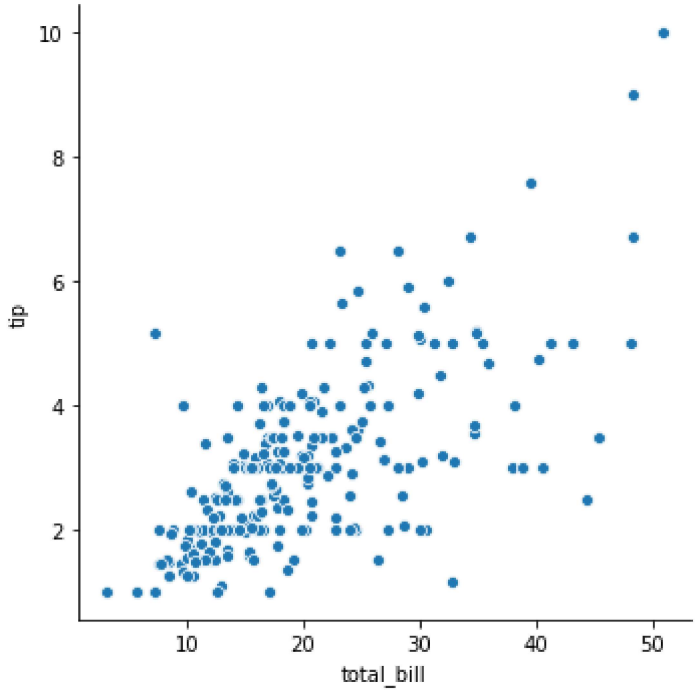
In [17]:
```
sns.scatterplot(x=df['total_bill'], y=df['tip'])
plt.show()
```



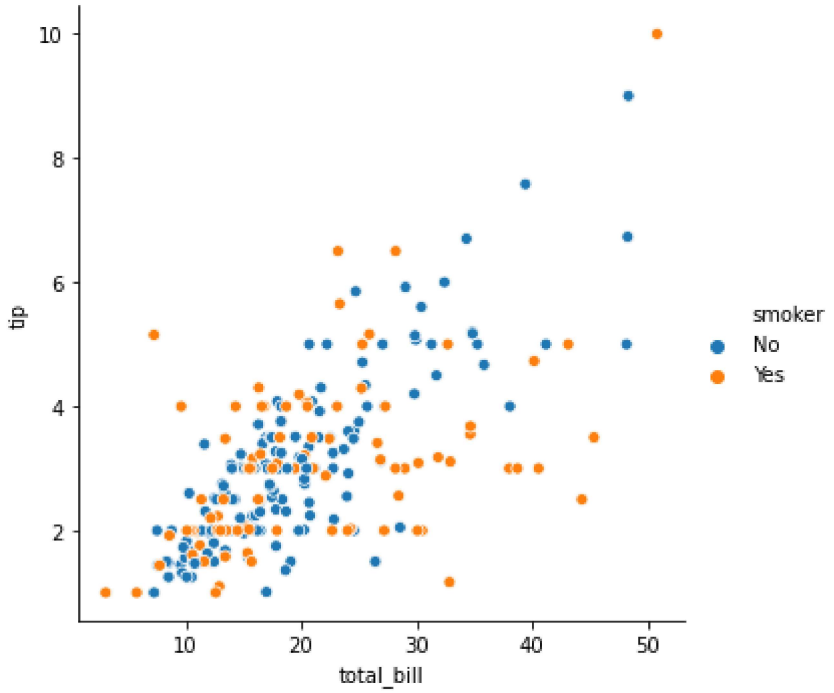
In [18]:
```
sns.scatterplot(x='total_bill', y='tip',data = df)
plt.show()
```



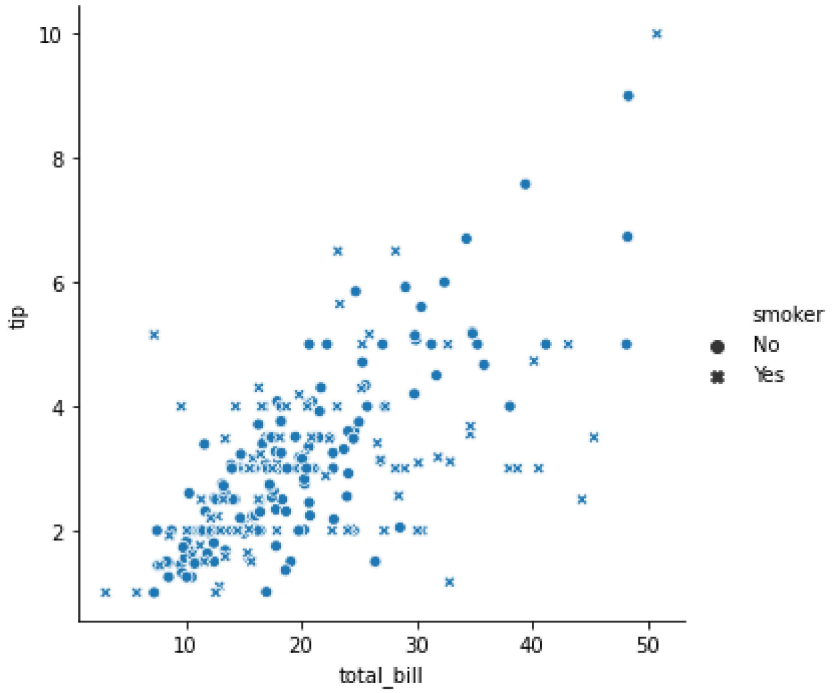
In [19]:
```
sns.relplot(x='total_bill', y='tip', data=df)
plt.show()
```



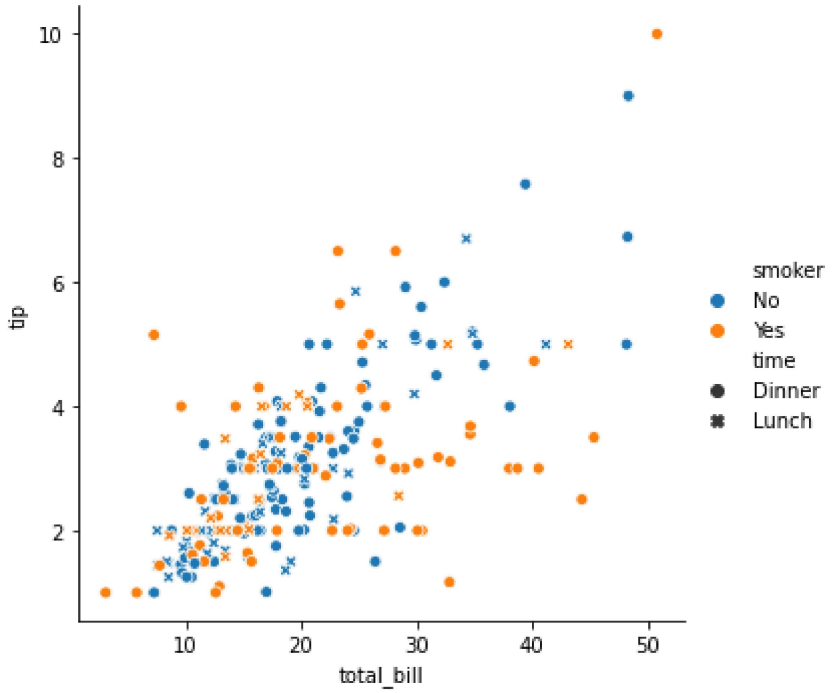
In [20]:
```
sns.relplot(x = 'total_bill', y = 'tip', data = df, hue = 'smoker')
plt.show()
```



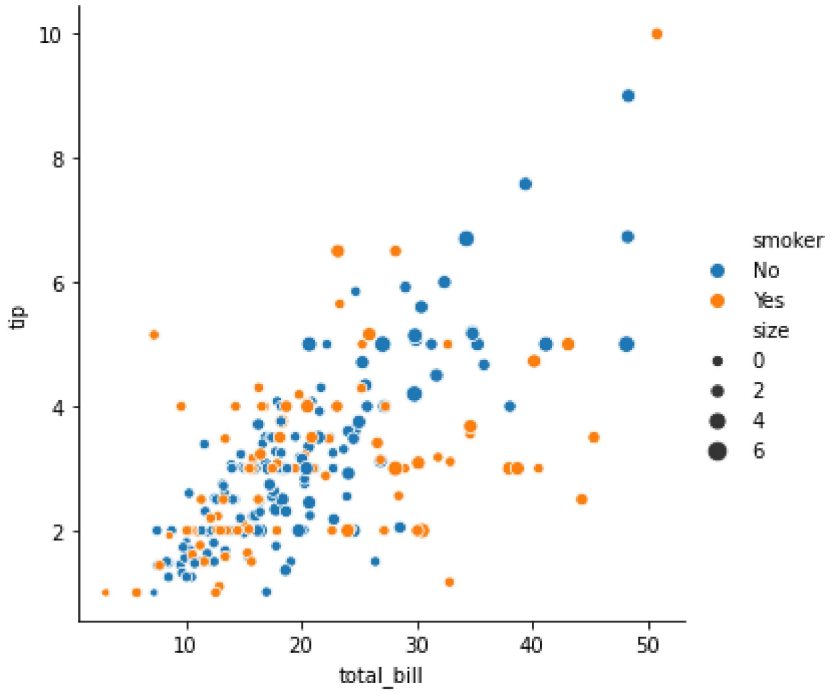
In [21]:
```
sns.relplot(x = 'total_bill', y = 'tip', data = df, style = 'smoker')
plt.show()
```



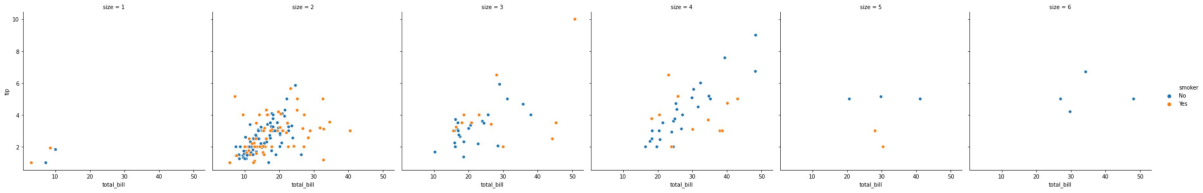
In [22]:
```
sns.relplot(x = 'total_bill', y = 'tip', data = df, hue = 'smoker', style = 't
ime')
plt.show()
```



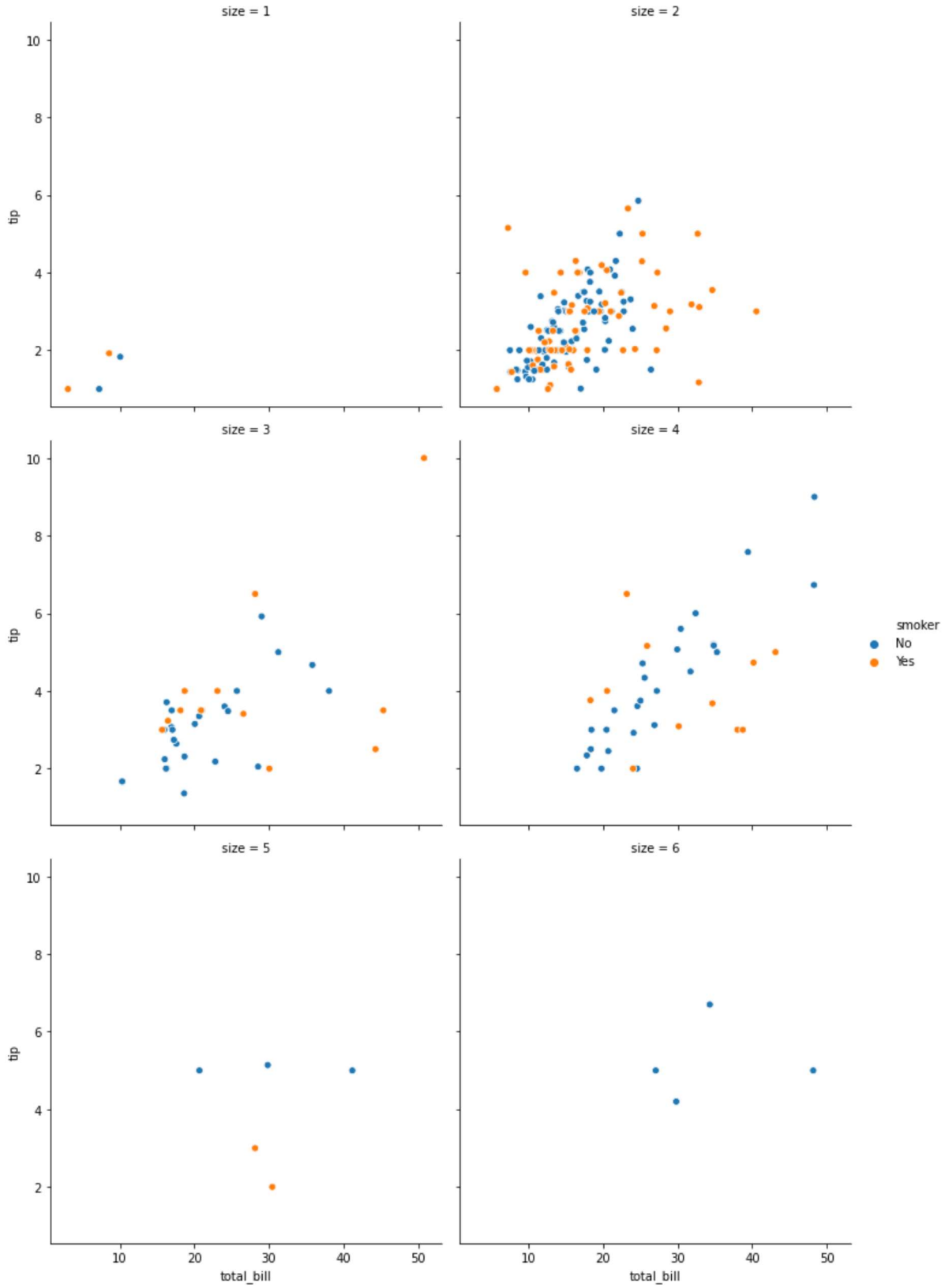
In [23]:
```
sns.relplot(x = 'total_bill', y = 'tip', data = df, hue = 'smoker', size= 'siz
e')
plt.show()
```



In [24]:
```
sns.relplot(x = 'total_bill', y = 'tip', data = df, hue = 'smoker', col = 'size')
plt.show()
```



In [25]:
```
sns.relplot(x = 'total_bill', y = 'tip',data = df, hue = 'smoker', col = 'siz
e',  col_wrap=2)
plt.show()
```



Line Plot

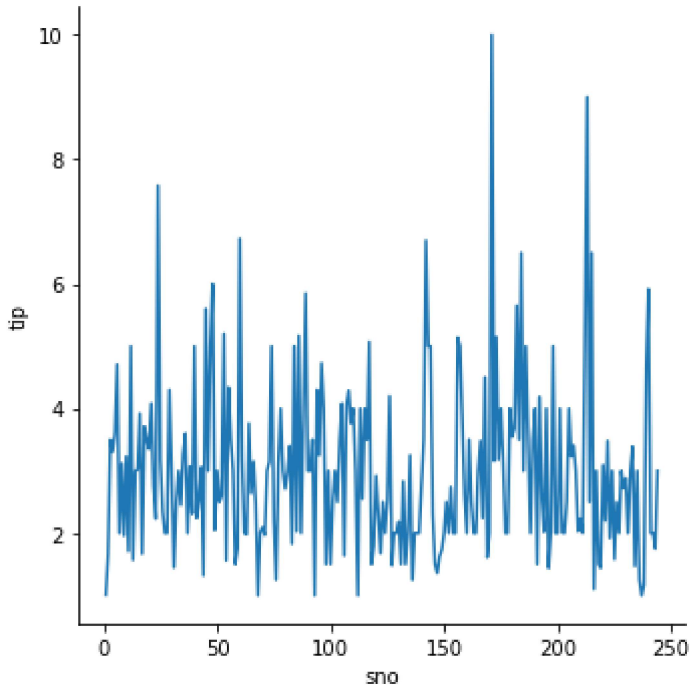
In [26]:
```
df['sno'] = pd.DataFrame(np.arange(1,245))
df
```

Out[26]:

	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

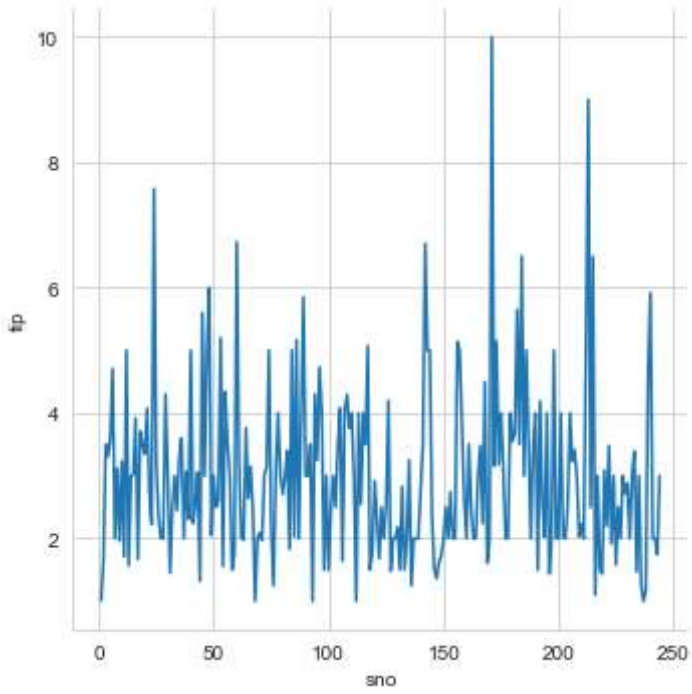
244 rows × 8 columns

In [27]:
```
sns.relplot(x = 'sno', y = 'tip', kind = 'line', data = df)
plt.show()
```



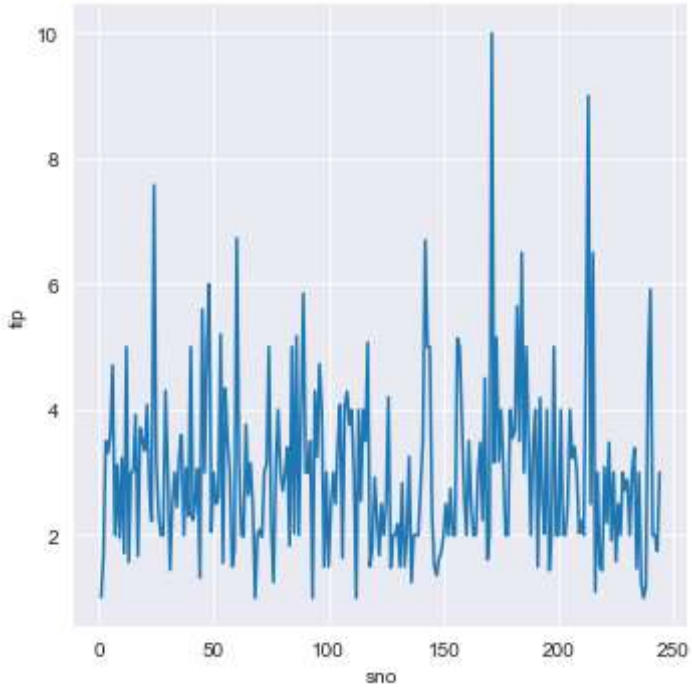
In [28]:
```
sns.set_style('whitegrid')
sns.relplot(x = 'sno', y = 'tip', kind = 'line', data = df)
```

Out[28]: <seaborn.axisgrid.FacetGrid at 0x27372550a90>



In [29]:
```
sns.set_style('darkgrid')
sns.relplot(x ='sno', y = 'tip', kind = 'line', data = df)
```

Out[29]: <seaborn.axisgrid.FacetGrid at 0x27372529bb0>

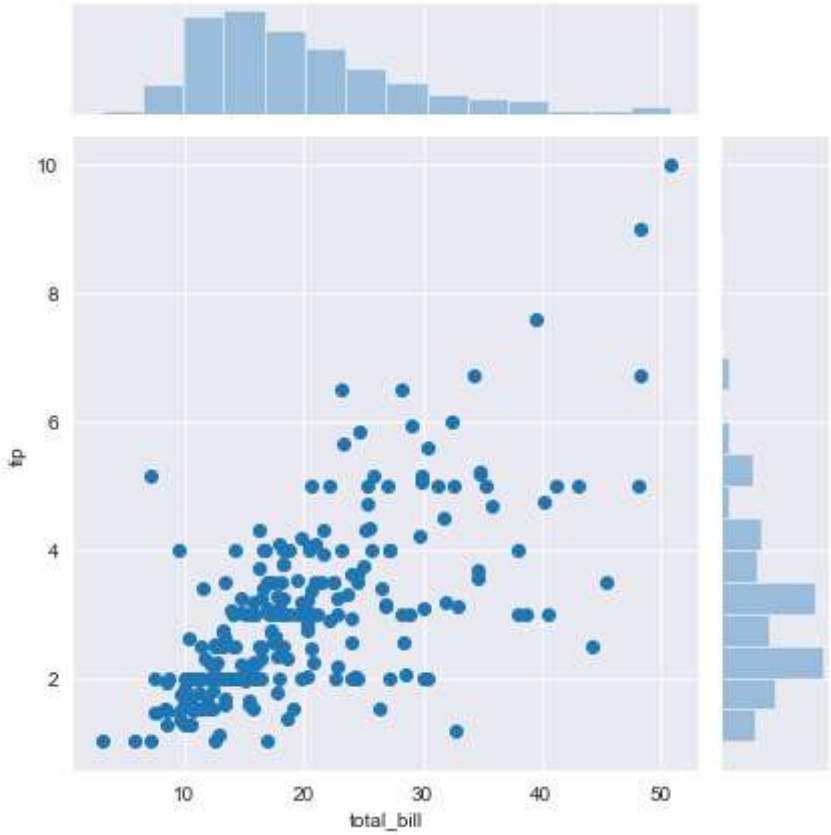


In [30]:
```
df.drop("sno",axis=1,inplace=True)
```

JoinPlot

A join plot allows to study the relationship between 2 numeric variables. The central chart display their correlation. It is usually a scatterplot, a hexbin plot, a 2D histogram or a 2D density plot

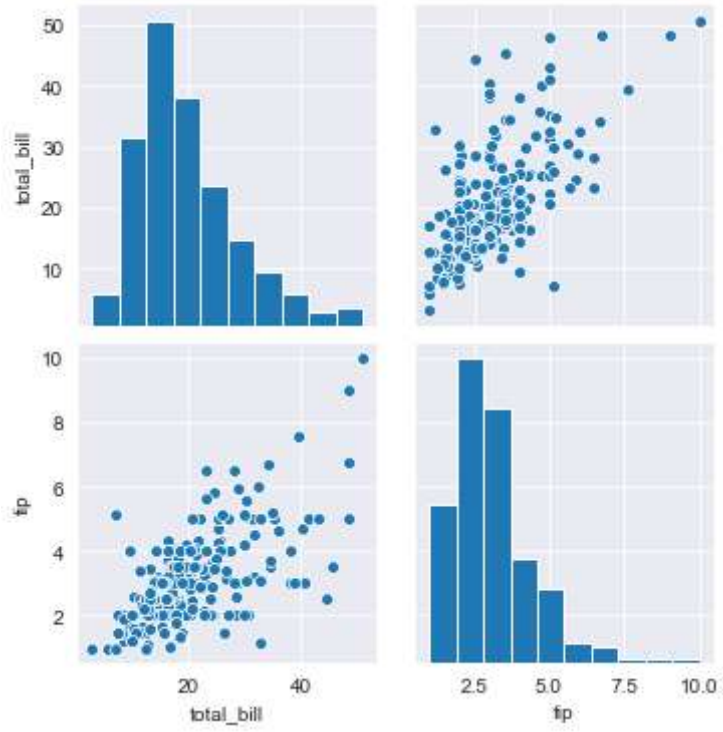
In [31]:
```
sns.jointplot(y='tip',x='total_bill',data=df)
plt.show()
```



Pair plot - Multiple continous variables

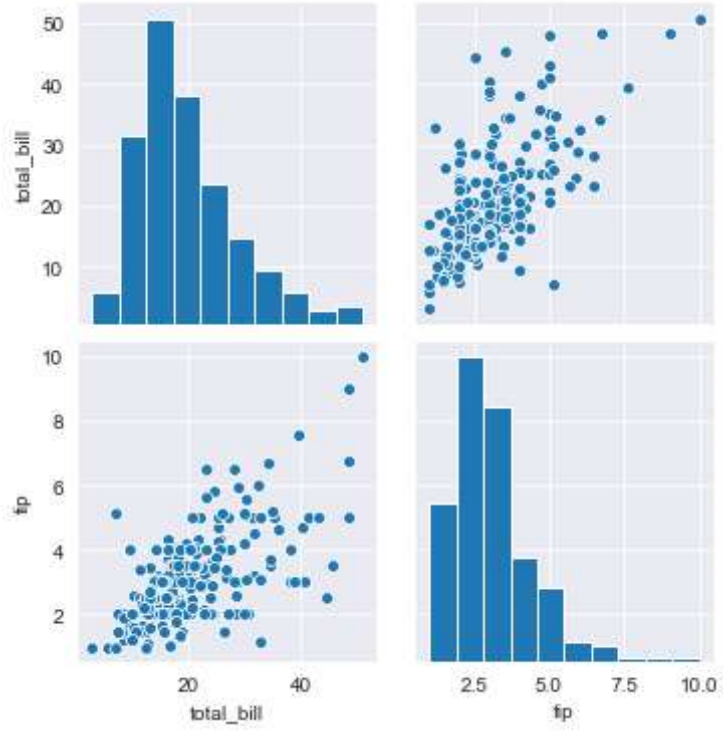
A “pairs plot” is also known as a scatterplot, in which one variable in the same data row is matched with another variable's value, like this: Pairs plots are just elaborations on this, showing all variables paired with all the other variables

In [32]:
```
sns.pairplot(df[['total_bill','tip']])
plt.show()
```



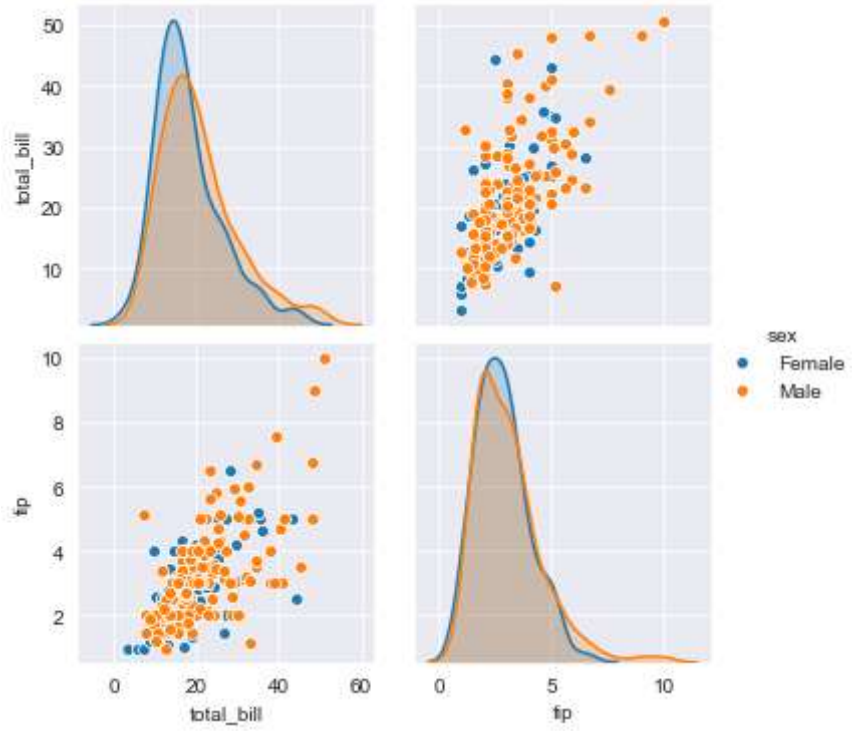
In [33]:
```
continous=['total_bill','tip']

sns.pairplot(df,vars=continous)
plt.show()
```



In [34]:
```
continous=['total_bill','tip']

sns.pairplot(df,vars=continous,hue='sex')
plt.show()
```



Heat Map

A heatmap uses colored cells to represent relation between variables

Heatmap (for correlation)

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

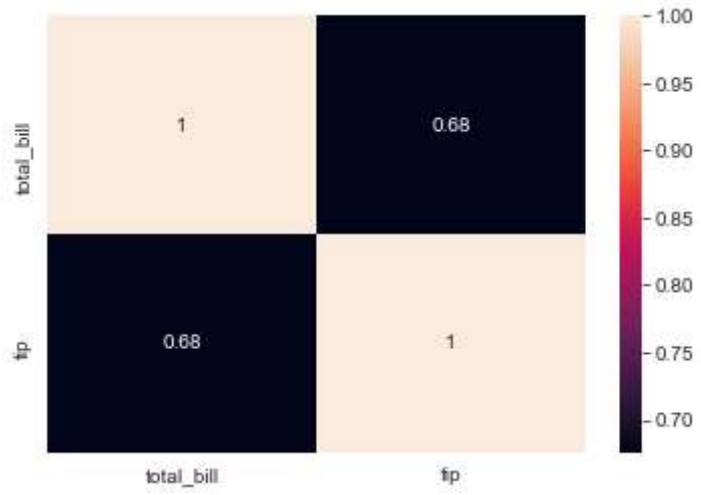
Correlation (r) = $\frac{COV(x,y)}{\sigma x \sigma y}$

In [35]:
```
corr_matrix = df[['total_bill','tip']].corr()
corr_matrix
```

Out[35]:

	total_bill	tip
total_bill	1.000000	0.675734
tip	0.675734	1.000000

In [36]:
```
sns.heatmap(corr_matrix,annot=True)
plt.show()
```



2. Plot's for Discrete Data

1. Univariate (Single Variable)
   - pie plot()
   - barplot()
   - countplot()

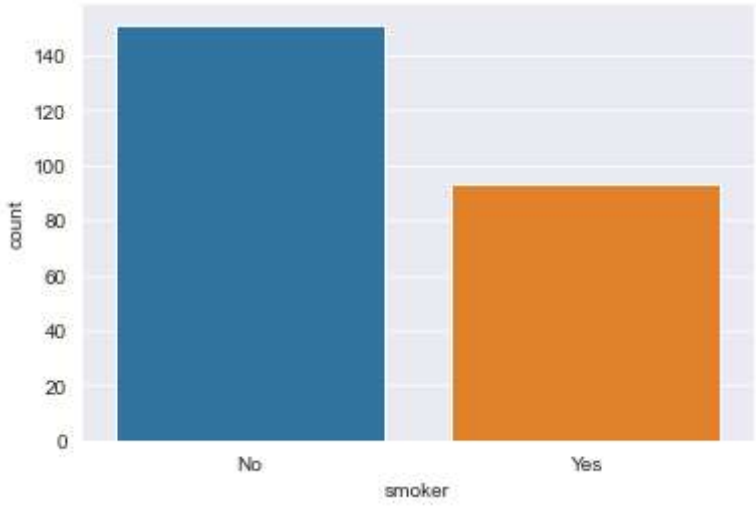
1. Bivariate (plot between two Variables)
   - boxplot() --> one discrete variable & one continous variable

CountPlot

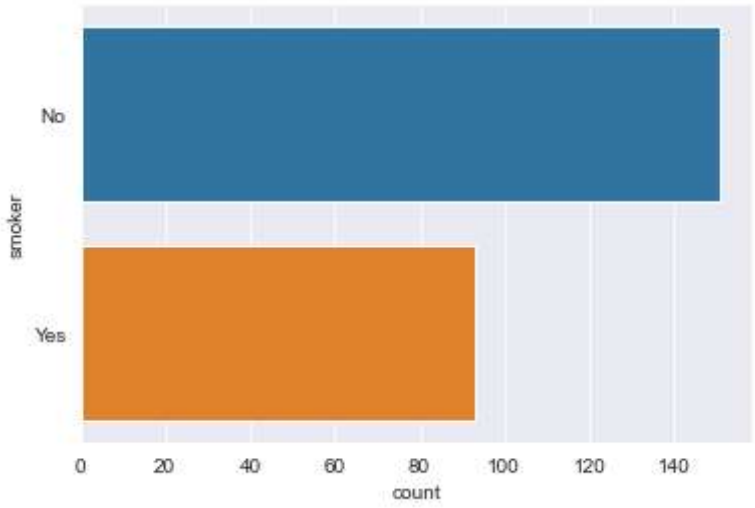
In [37]:
```
df['smoker'].value_counts()
```

Out[37]:
```
No     151
Yes     93
Name: smoker, dtype: int64
```

In [38]:
```
sns.countplot(x='smoker',data=df)
plt.show()
```

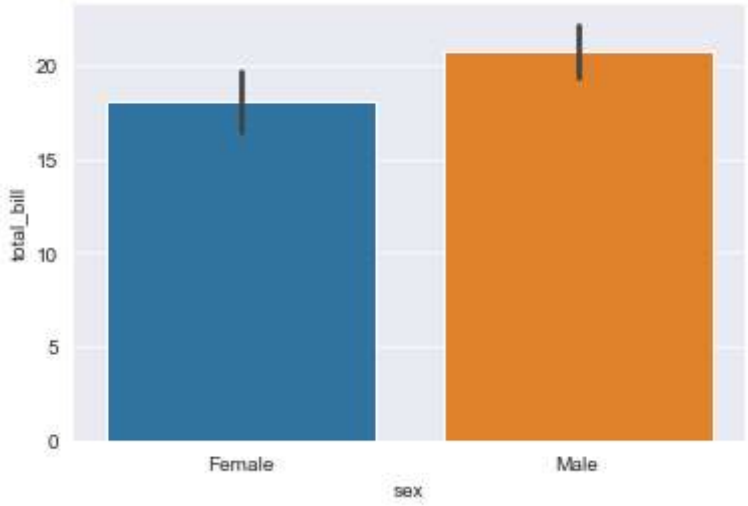


In [39]:
```
sns.countplot(y='smoker',data=df)
plt.show()
```

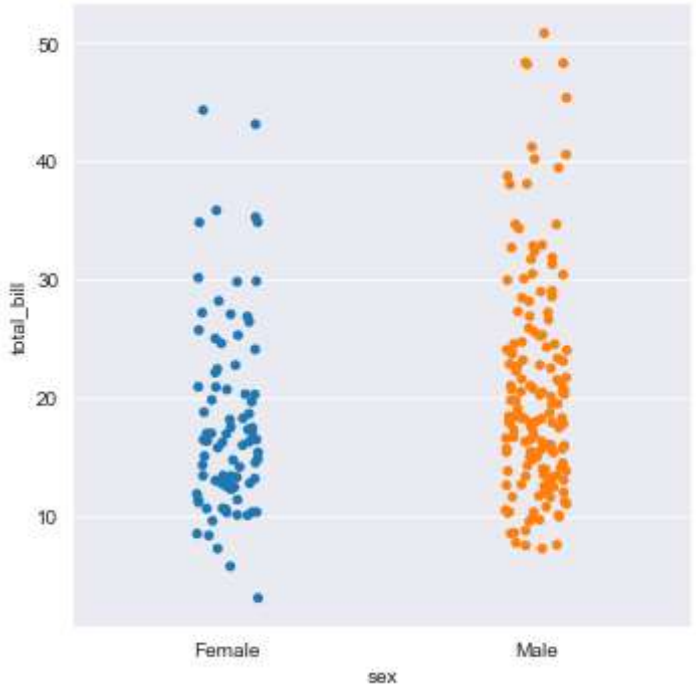


Bar plot

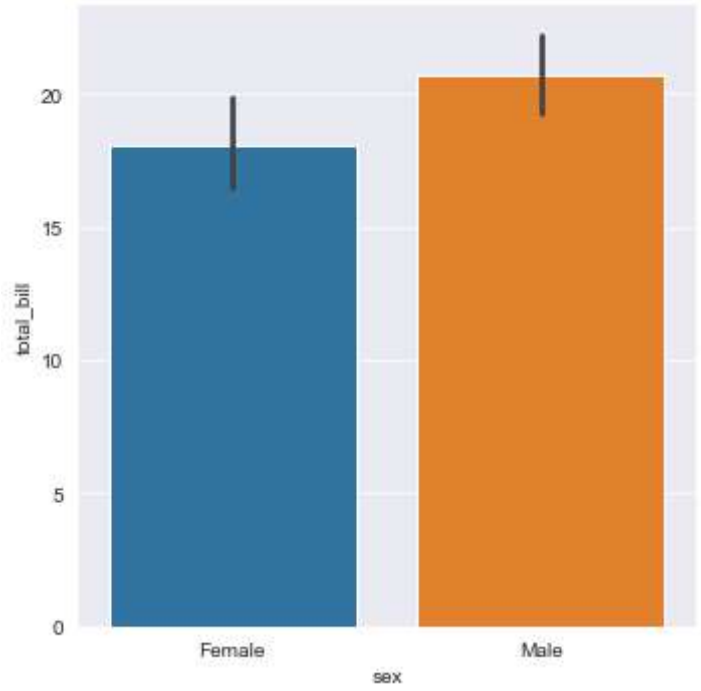
In [40]:
```
sns.barplot(x='sex',y='total_bill',data=df)
plt.show()
```



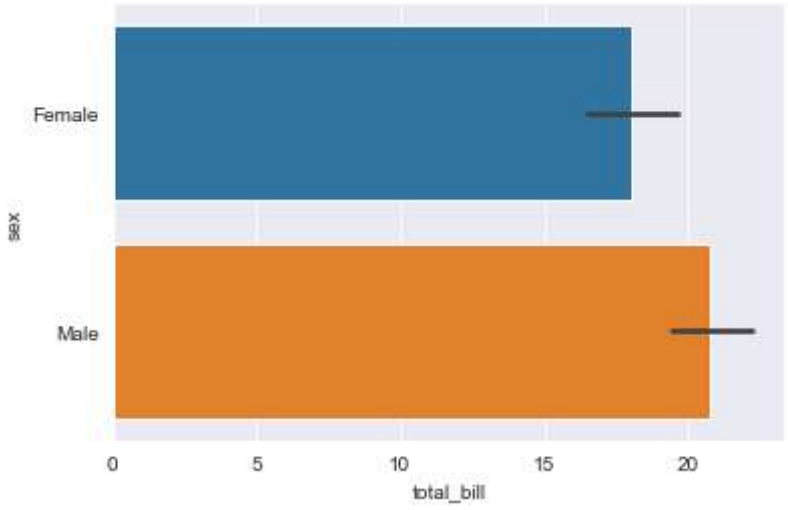
In [41]:
```
sns.catplot(x='sex',y='total_bill',data=df)
plt.show()
```



In [42]:
```
sns.catplot(x='sex',y='total_bill',data=df, kind="bar")
plt.show()
```



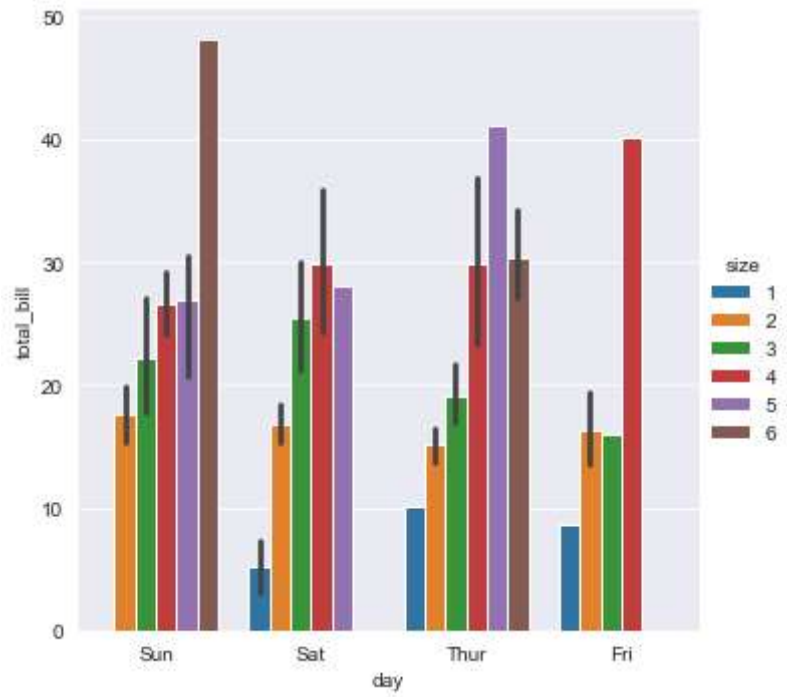
In [43]:
```
sns.barplot(x='total_bill',y='sex',data=df)
plt.show()
```



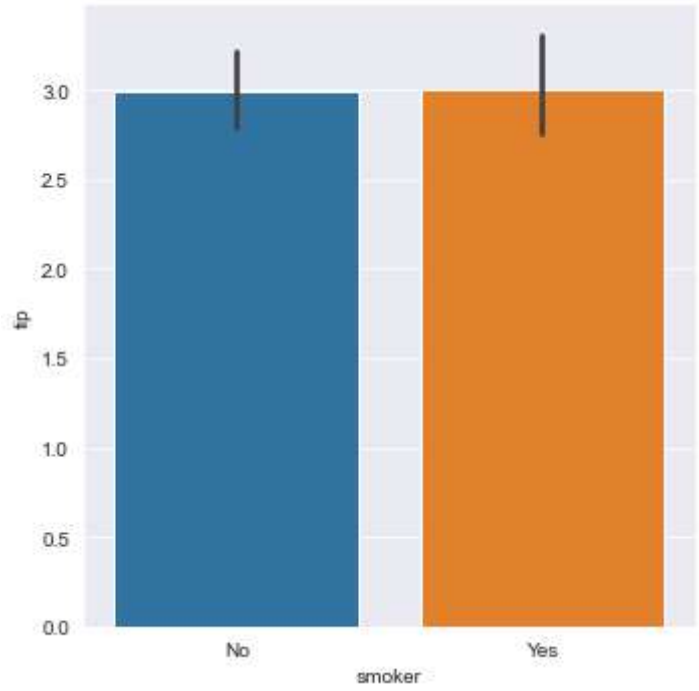
In [44]:
```
df['size'].value_counts()
```

Out[44]:
```
2    156
3     38
4     37
5      5
6      4
1      4
Name: size, dtype: int64
```

In [45]:
```
sns.catplot(x = 'day', y = 'total_bill', data = df, kind="bar", hue = 'size')
plt.show()
```

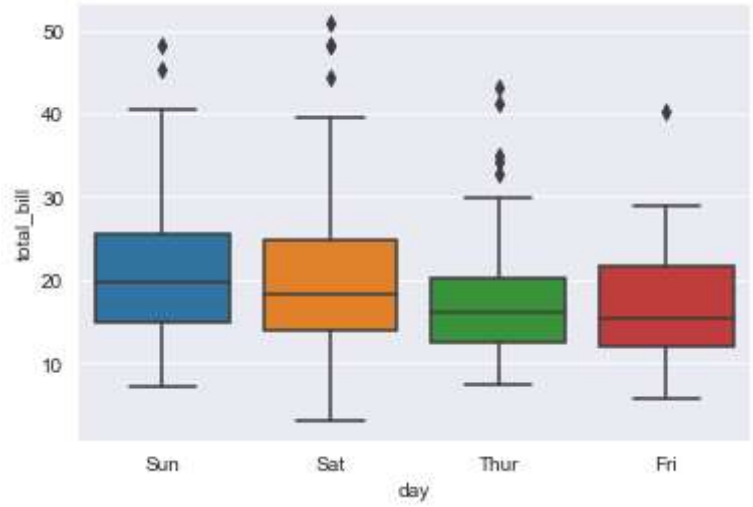


In [46]:
```
sns.catplot(x = 'smoker', y = 'tip', data = df, kind="bar",order = ['No', 'Yes'])
plt.show()
```

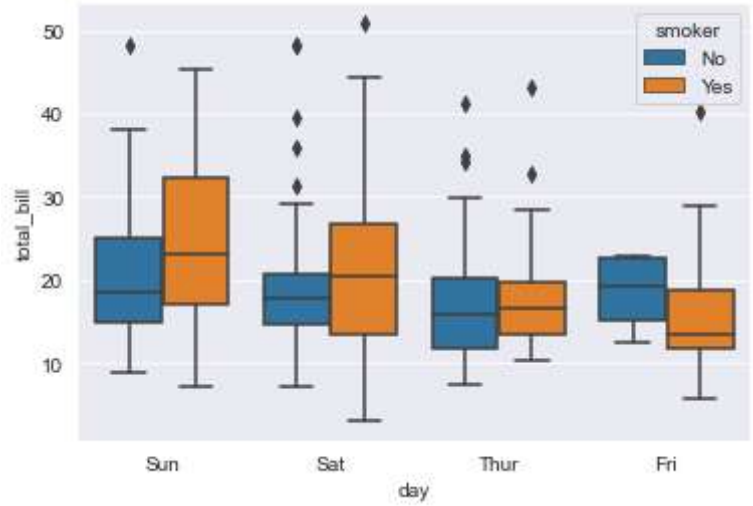


Box plot

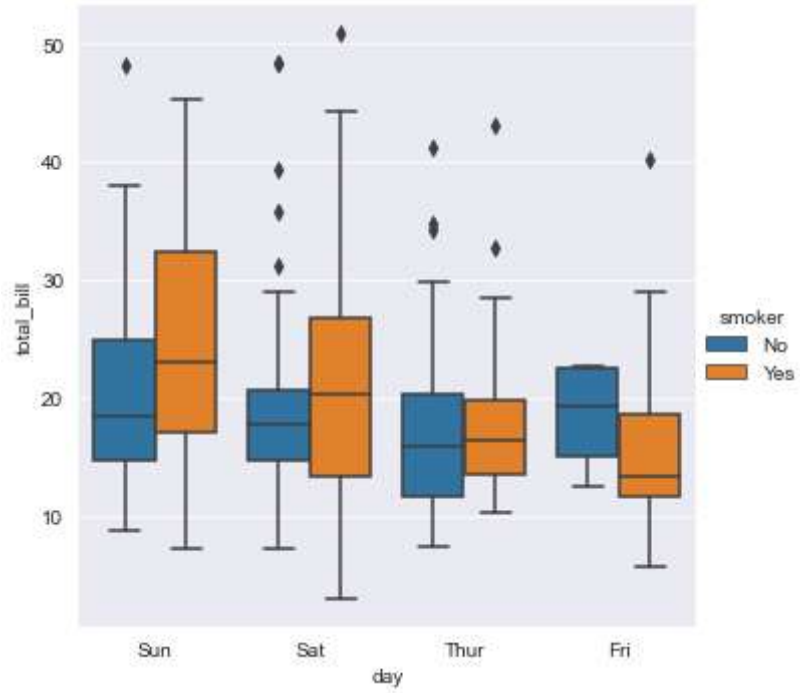
In [47]:
```
sns.boxplot(x="day", y="total_bill", data=df)
plt.show()
```



In [48]:
```
sns.boxplot(x="day", y="total_bill", hue="smoker", data=df)
plt.show()
```



In [49]:
```
sns.catplot(y="total_bill", x="day", hue="smoker", data=df, kind = 'box')
plt.show()
```



Violin Plot

Violin plot helps us to see both the distribution of data in terms of Kernel density estimation and the box plot

In [50]:
```
sns.violinplot(x="total_bill", y="day",data=df,palette='rainbow')
plt.show()
```