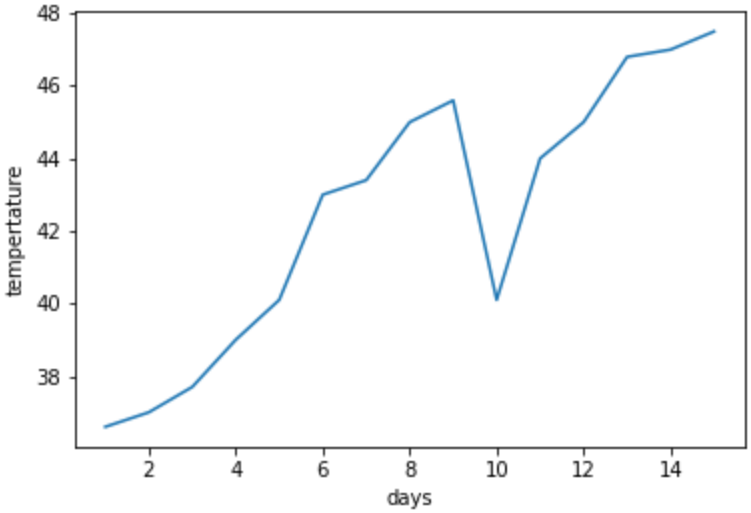
In [1]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

Line Plot

In [2]:
```
days=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
tempertature=[36.6,37,37.7,39,40.1,43,43.4,45,45.6,40.1,44,45,46.8,47,47.5]

df=({"days":days,"tempertature":tempertature})#creating data frame
sns.lineplot(x="days",y='tempertature',data=df)#to generate x and y axis names we provide
plt.show()                                                    #and data is taken by dict comp
```



In [3]:
```
tips_df=sns.load_dataset("tips")#Directly load from github
tips_df
```

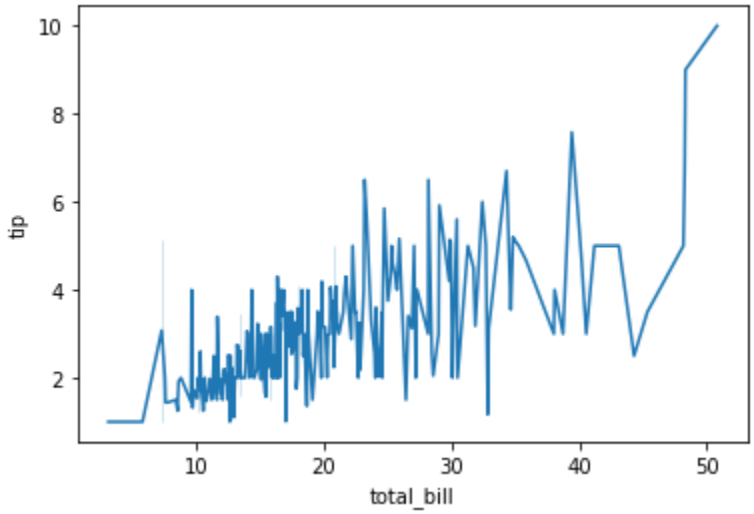
Out[3]:

	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
...	...	...	...	...	...	...	...
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

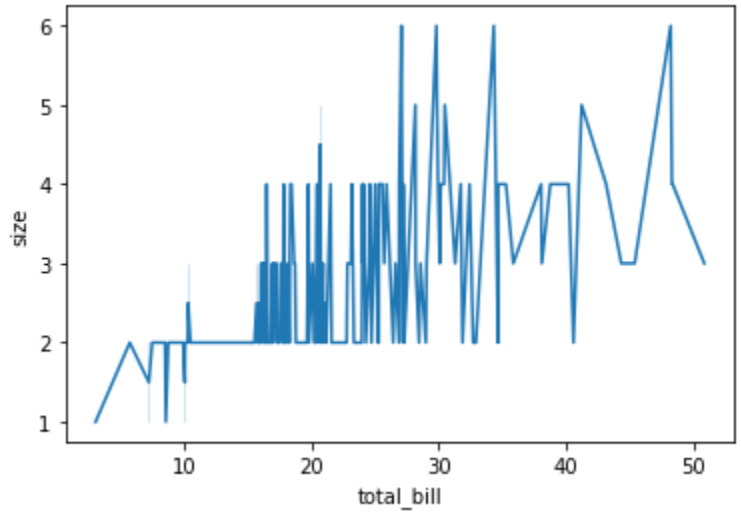
In [4]:
```
sns.lineplot(x="total_bill",y="tip",data=tips_df)
```

Out[4]: <AxesSubplot:xlabel='total_bill', ylabel='tip'>

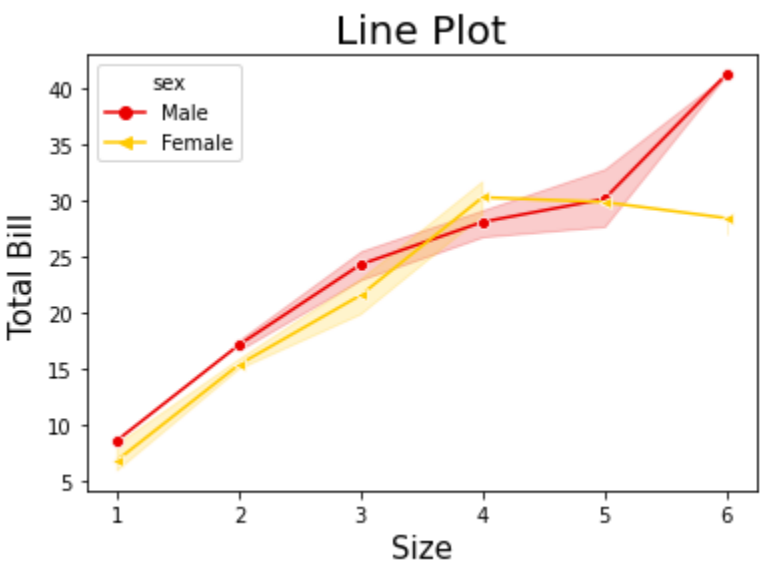


In [5]:
```
sns.lineplot(x="total_bill",y="size",data=tips_df)
```

Out[5]: <AxesSubplot:xlabel='total_bill', ylabel='size'>



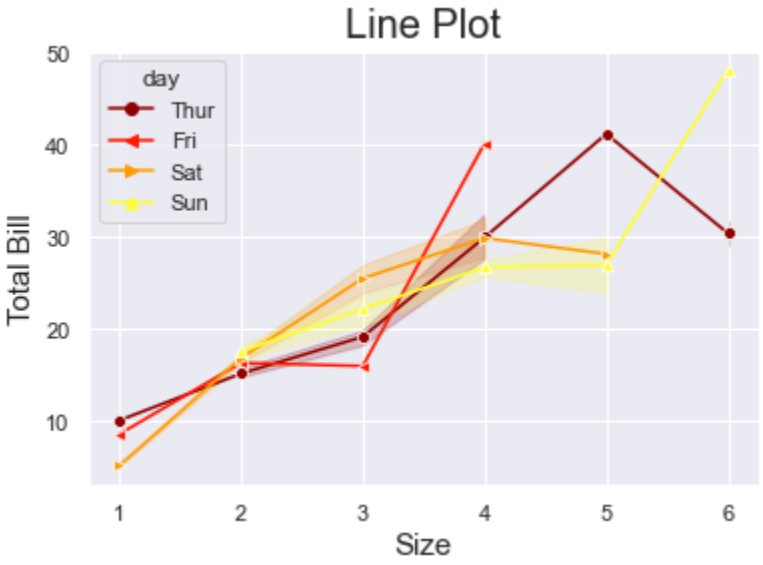
In [6]:
```
sns.lineplot(x="size",y="total_bill",data=tips_df,hue="sex",
             style='sex',palette='hot',dashes=False,
             markers=['o','<'],legend='auto',ci=50)
#hue is used to plot based on given column name and generated automatic legend

#ci : int or "sd" or None-> Jitna kam values utni kam standard deviation
    #Size of the confidence interval to draw when aggregating with an
    #estimator. "sd" means to draw the standard deviation of the data

plt.title("Line Plot",fontsize=20)
plt.xlabel("Size",fontsize=15)
plt.ylabel("Total Bill",fontsize=15)
plt.show()
sns.set(style="darkgrid",)#To apply grid
plt.figure(figsize=(10,9))
```



Out[6]:
```
<Figure size 720x648 with 0 Axes>
```


```
<Figure size 720x648 with 0 Axes>
```


In [7]:
```
sns.lineplot(x="size",y="total_bill",data=tips_df,hue="day",
             style='day',palette='hot',dashes=False,
             markers=['o','<','>','^'],legend='auto',ci=50)
#hue is used to plot based on given column name and generated automatic legend

#ci : int or "sd" or None-> Jitna kam values utni kam standard deviation
    #Size of the confidence interval to draw when aggregating with an
    #estimator. "sd" means to draw the standard deviation of the data

plt.title("Line Plot",fontsize=20)
plt.xlabel("Size",fontsize=15)
plt.ylabel("Total Bill",fontsize=15)
plt.show()
sns.set(style="darkgrid",)#To apply grid
plt.figure(figsize=(10,9))
```



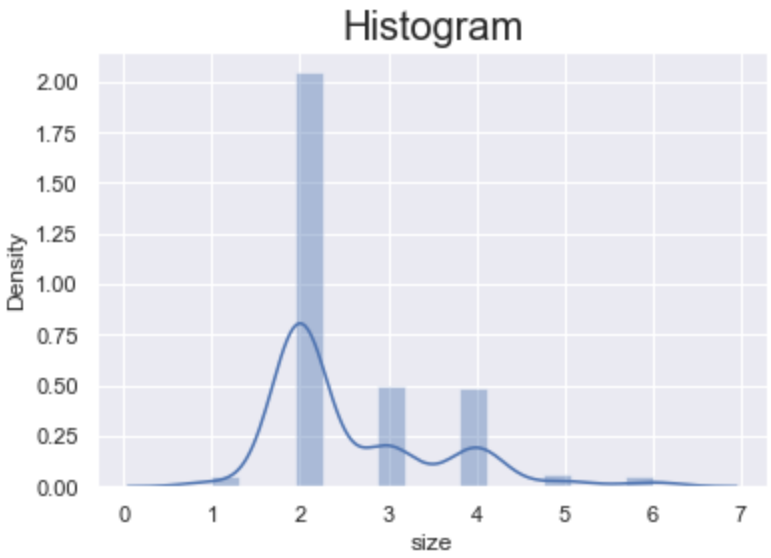
Out[7]:
```
<Figure size 720x648 with 0 Axes>
```


```
<Figure size 720x648 with 0 Axes>
```


In [8]:
```
sns.distplot(tips_df['size']) #here line in chart is known as KDE(Kernel Density Extimati
                              #(KDE) is a non-parametric way to estimate the probability de
plt.title("Histogram",fontsize=20)
```

```
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 adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

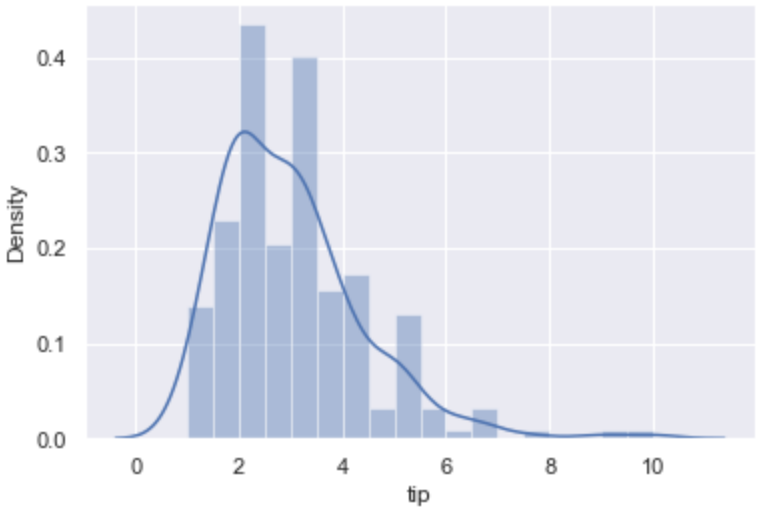
Out[8]:
```
Text(0.5, 1.0, 'Histogram')
```



In [9]:
```
sns.distplot(tips_df['tip'])
```

```
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 adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

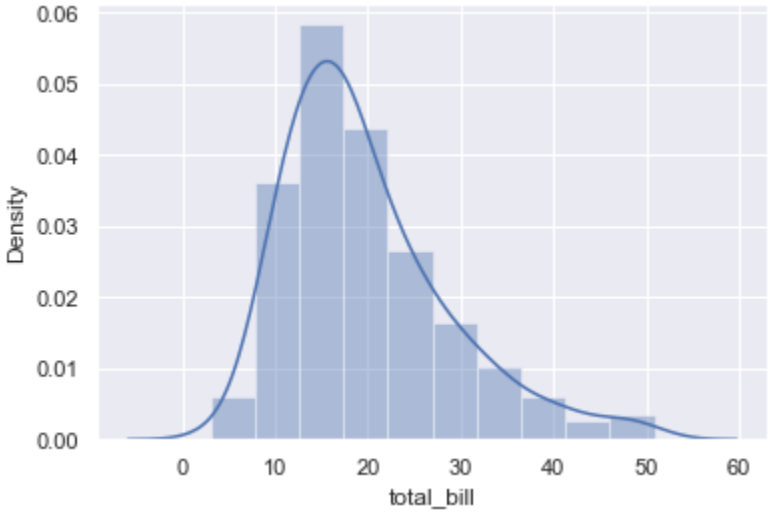
Out[9]:
```
<AxesSubplot:xlabel='tip', ylabel='Density'>
```



In [10]:
```
sns.distplot(tips_df['total_bill'],bins=10)#bins used to discribe data in specified no. h
```

```
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 adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

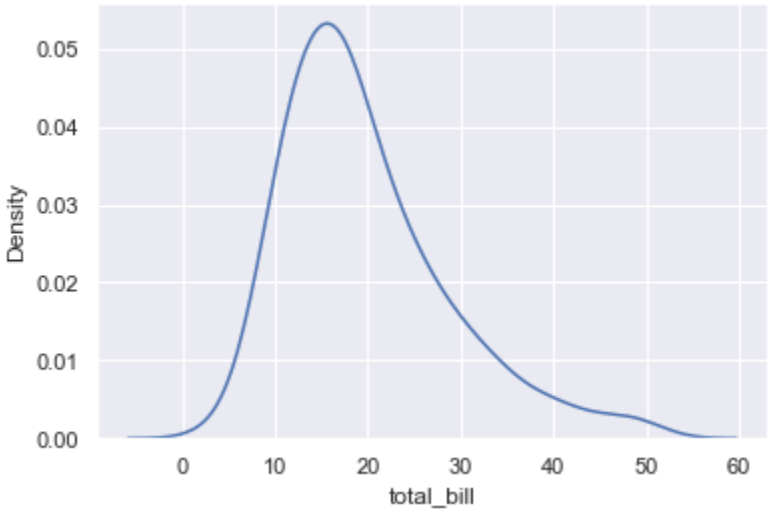
Out[10]:
```
<AxesSubplot:xlabel='total_bill', ylabel='Density'>
```



In [11]:
```
sns.distplot(tips_df['total_bill'],bins=100,hist=False)
```

```
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 adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

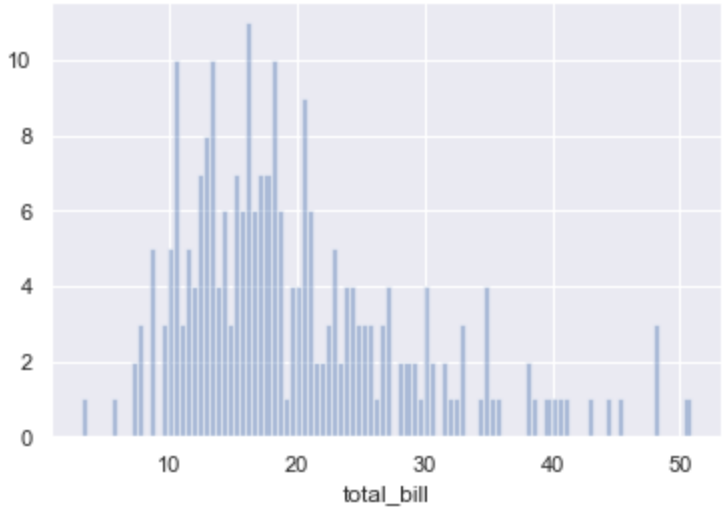
Out[11]:
```
<AxesSubplot:xlabel='total_bill', ylabel='Density'>
```



In [12]:
```
sns.distplot(tips_df['total_bill'],bins=100,hist=True,kde=False)
```

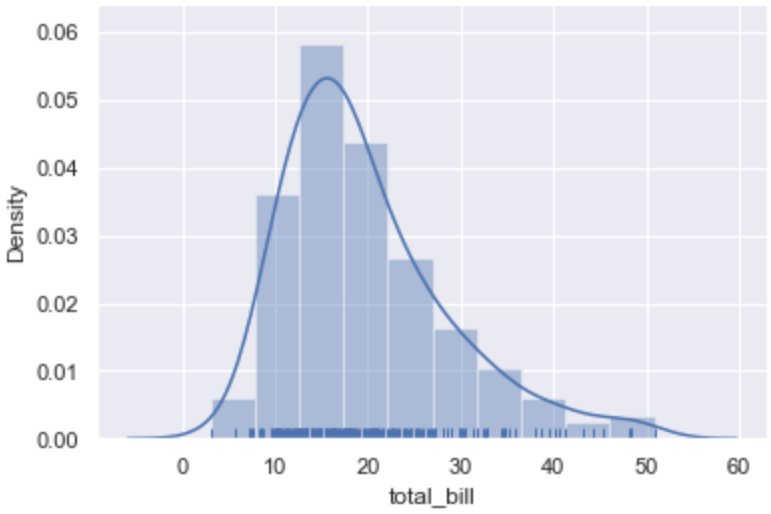
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function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

Out[12]:
```
<AxesSubplot:xlabel='total_bill'>
```



In [13]:
```
sns.distplot(tips_df['total_bill'],bins=10,rug=True)#A rugplot is a graph that places a d
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2103: FutureWarning: T
he `axis` variable is no longer used and will be removed. Instead, assign variables direct
ly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

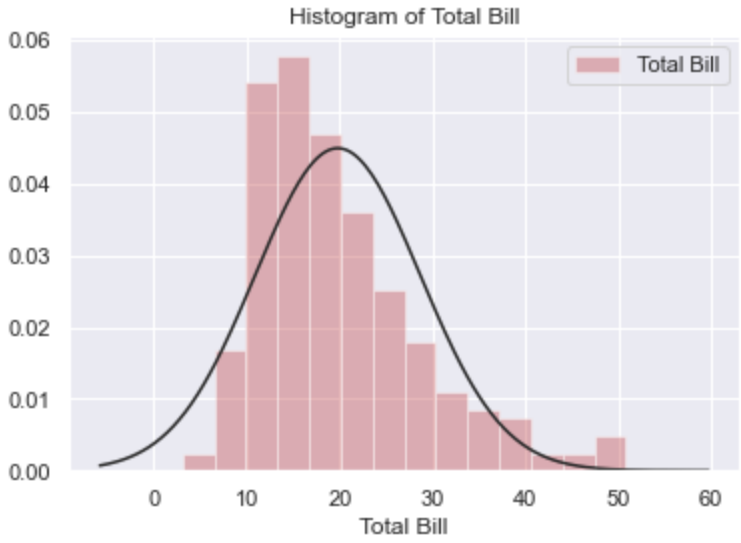


In [14]:
```
from scipy.stats import norm
```

In [15]:
```
sns.distplot(tips_df['total_bill'],fit=norm,kde=False,color='r',axlabel='Total Bill',labe
#norm is normalised/gaussian distribution fit
plt.legend()
plt.title("Histogram of Total Bill")
```

```
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 adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

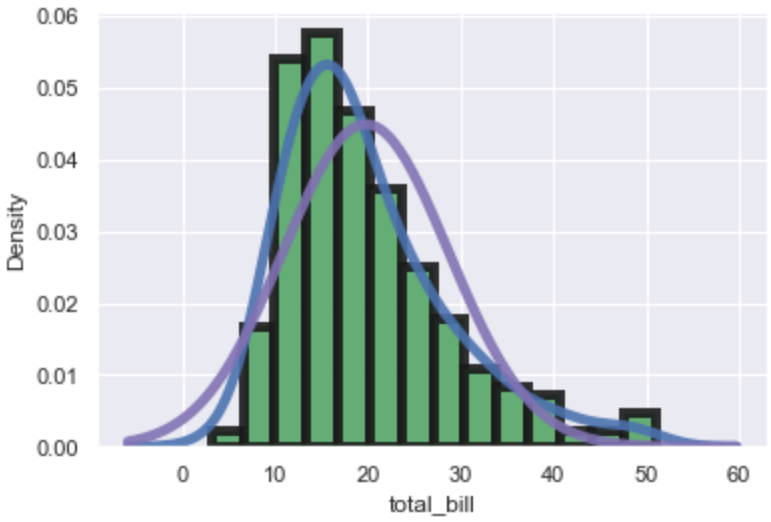
Out[15]:
```
Text(0.5, 1.0, 'Histogram of Total Bill')
```



In [16]:

```
sns.distplot(tips_df['total_bill'],hist_kws={'color':'g','linewidth':5,'edgecolor':'k','a
             kde_kws={'color':'b','linewidth':5,'alpha':0.9},
             fit=norm,fit_kws={'color':'m','linewidth':5,'alpha':0.9})
#hist_kws used to pass diff arguments in key-value pair
#kde_kws has no edgecolor support
sns.set()
```

```
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 adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [17]:

```
tips_df=sns.load_dataset("tips")#Directly load from github
tips_df
```

Out[17]:

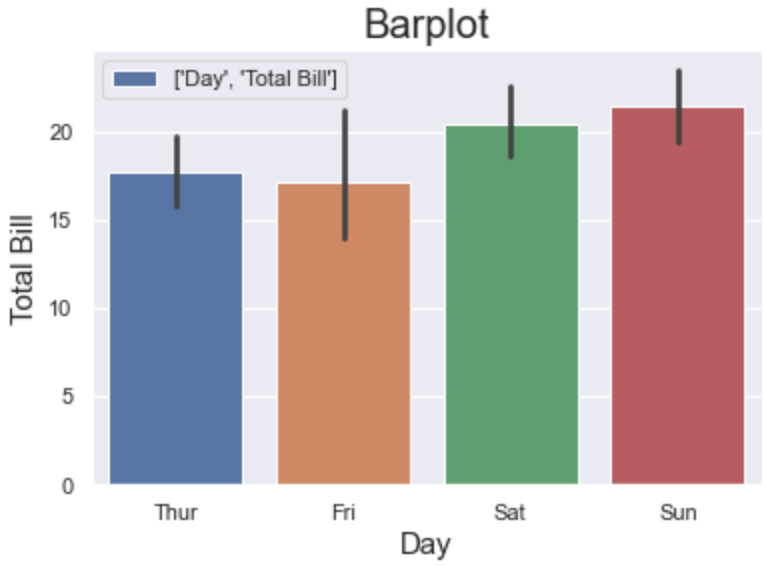
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...

	total_bill	tip	sex	smoker	day	time	size
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

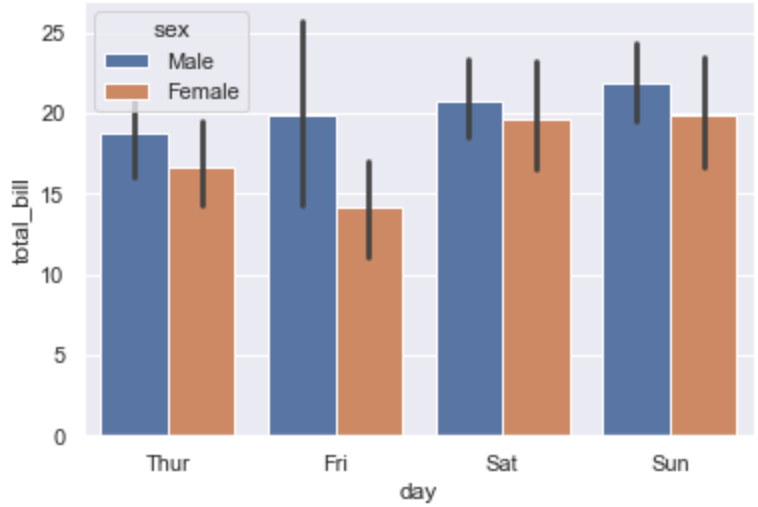
In [18]:
```
sns.barplot(x=tips_df.day,y=tips_df.total_bill,label=['Day','Total Bill'])#agr data nhi u
plt.title('Barplot',fontsize=20)
plt.xlabel('Day',fontsize=15)
plt.ylabel('Total Bill',fontsize=15)
plt.legend()
```

Out[18]:
```
<matplotlib.legend.Legend at 0x1aaf2319070>
```



In [19]:
```
sns.barplot(x='day',y='total_bill',hue='sex',data=tips_df,alpha=1)
```

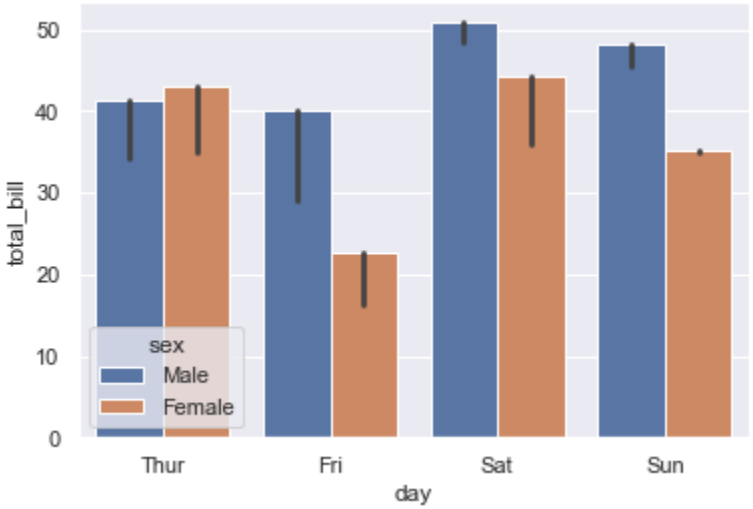
Out[19]:
```
<AxesSubplot:xlabel='day', ylabel='total_bill'>
```



In [20]:
```
np
```

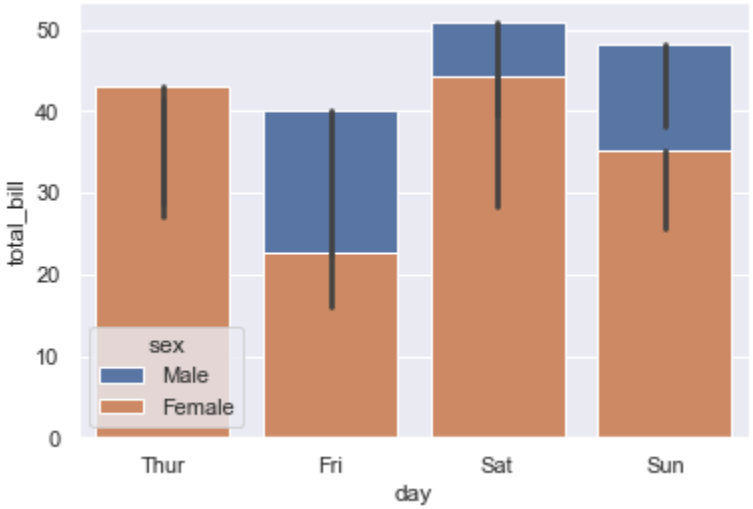
In [21]:
```
sns.barplot(x='day',y='total_bill',hue='sex',data=tips_df,estimator=np.max,ci=70)
#A confidence interval displays the probability that a parameter will fall
#between a pair of values around the mean
#estimator used to change the y-axis values
```

Out[21]:
```
<AxesSubplot:xlabel='day', ylabel='total_bill'>
```

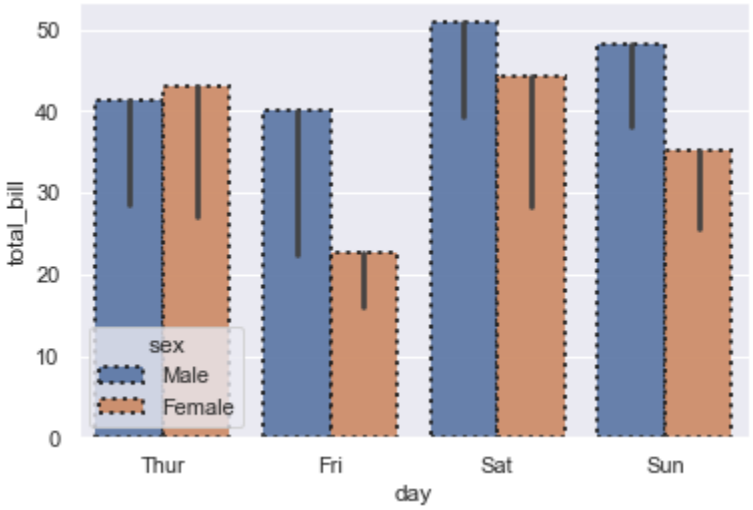


In [22]:
```
sns.barplot(x='day',y='total_bill',hue='sex',data=tips_df,estimator=np.max,dodge=False)
```

Out[22]:
```
<AxesSubplot:xlabel='day', ylabel='total_bill'>
```



In [23]:
```
kwargs={'alpha':0.9,'linestyle':':','linewidth':2,'edgecolor':'k'}
sns.barplot(x='day',y='total_bill',hue='sex',data=tips_df,estimator=np.max,**kwargs)
sns.set()
```



In [24]:
```
df=sns.load_dataset('titanic')
df
```

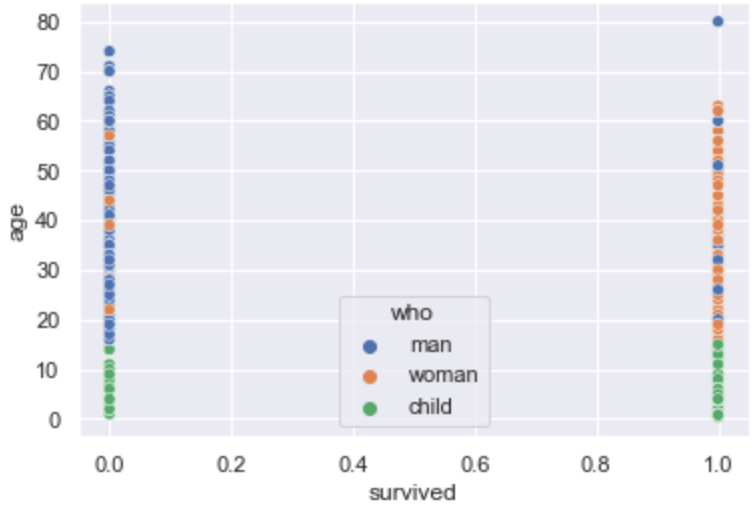
Out[24]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	d
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	
...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	

891 rows × 15 columns

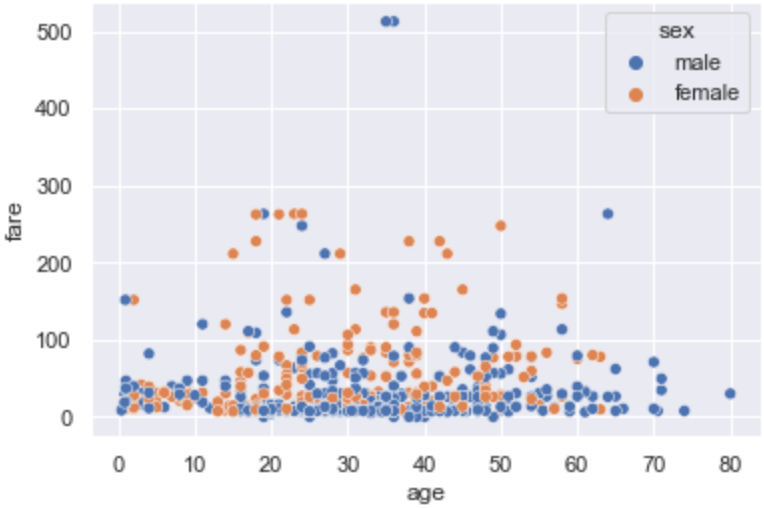
In [25]:
```
sns.scatterplot(x=df.survived,y=df.age,hue=df.who)
```

Out[25]:
```
<AxesSubplot:xlabel='survived', ylabel='age'>
```



In [26]:
```
sns.scatterplot(x='age',y='fare',hue='sex',data=df)
```

Out[26]:
```
<AxesSubplot:xlabel='age', ylabel='fare'>
```

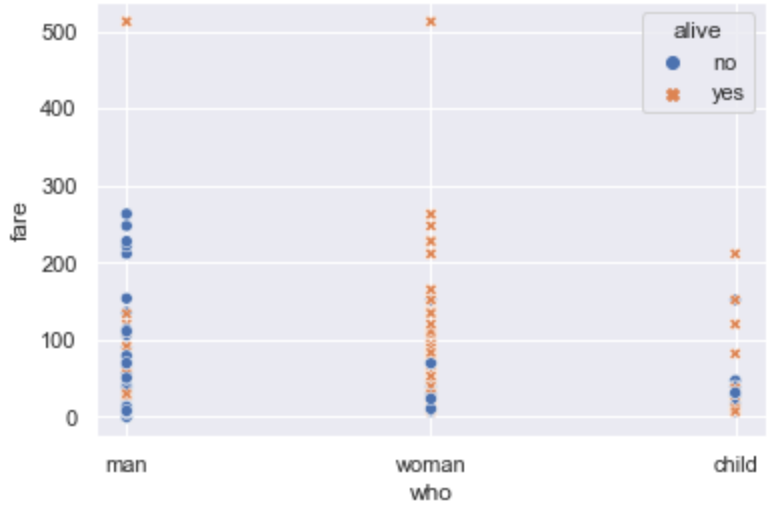


In [27]:
```
sns.scatterplot(x='age',y='fare',hue='sex',data=df,style='who')
```

Out[27]:
```
<AxesSubplot:xlabel='age', ylabel='fare'>
```

In [28]:
```
sns.scatterplot(x='who',y='fare',hue='alive',data=df,style='alive')
plt.figure(figsize=(16,10))
```

Out[28]:
```
<Figure size 1152x720 with 0 Axes>
```




```
<Figure size 1152x720 with 0 Axes>
```

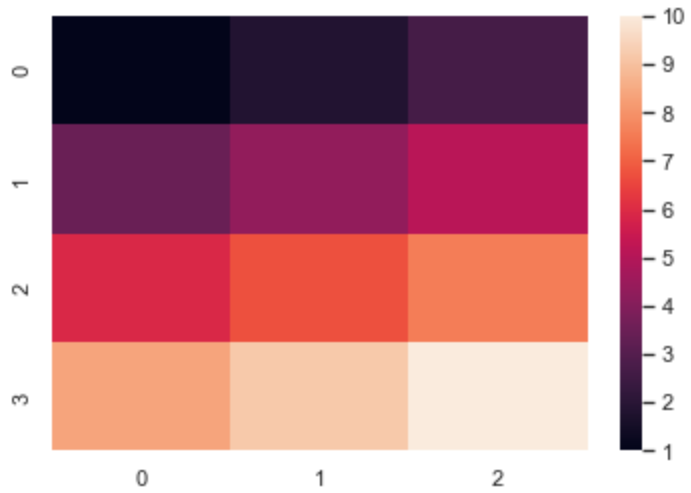
Heatmap used to represent 2D data

In [29]:
```
arr=np.linspace(1,10,12).reshape(4,3)
arr
```

Out[29]:
```
array([[ 1.        ,  1.81818182,  2.63636364],
       [ 3.45454545,  4.27272727,  5.09090909],
       [ 5.90909091,  6.72727273,  7.54545455],
       [ 8.36363636,  9.18181818, 10.        ]])
```

In [30]:
```
sns.heatmap(arr)
```

Out[30]:
```
<AxesSubplot:>
```



In [31]:
```
# Load dataset

globalWarming_df = pd.read_csv("E:\\New folder\\Who_is_responsible_for_global_warming.csv
globalWarming_df
```

Out[31]:

	Country Name	Country Code	Indicator Name	Indicator Code	2000	2001	2002	2003

	Country Name	Country Code	Indicator Name	Indicator Code	2000	2001	2002	2003	
0	United States	USA	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	20.178751	19.636505	19.613404	19.564105	19.65
1	United Kingdom	GBR	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	9.199549	9.233175	8.904123	9.053278	8.98
2	India	IND	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	0.979870	0.971698	0.967381	0.992392	1.02
3	China	CHN	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	2.696862	2.742121	3.007083	3.524074	4.03
4	Russian Federation	RUS	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	10.627121	10.669603	10.715901	11.090647	11.12
5	Australia	AUS	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	17.200610	16.733367	17.370452	16.901959	17.02
6	France	FRA	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	5.946665	6.153061	6.068664	6.115998	6.12
7	Germany	DEU	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	10.095640	10.366287	10.058673	9.969355	9.89
8	Canada	CAN	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	17.367115	16.985030	16.559378	17.461199	17.25
9	Brazil	BRA	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	1.871118	1.898354	1.844380	1.762482	1.82
10	Argentina	ARG	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	3.835574	3.568600	3.291548	3.525584	4.06
11	Pakistan	PAK	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	0.768458	0.764702	0.788668	0.804959	0.87

	Country Name	Country Code	Indicator Name	Indicator Code	2000	2001	2002	2003	
12	Nepal	NPL	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	0.129282	0.135226	0.106877	0.113902	0.1(
13	Bangladesh	BGD	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	0.211802	0.242020	0.246756	0.256602	0.26
14	Japan	JPN	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	9.622352	9.464309	9.573130	9.725282	9.9(

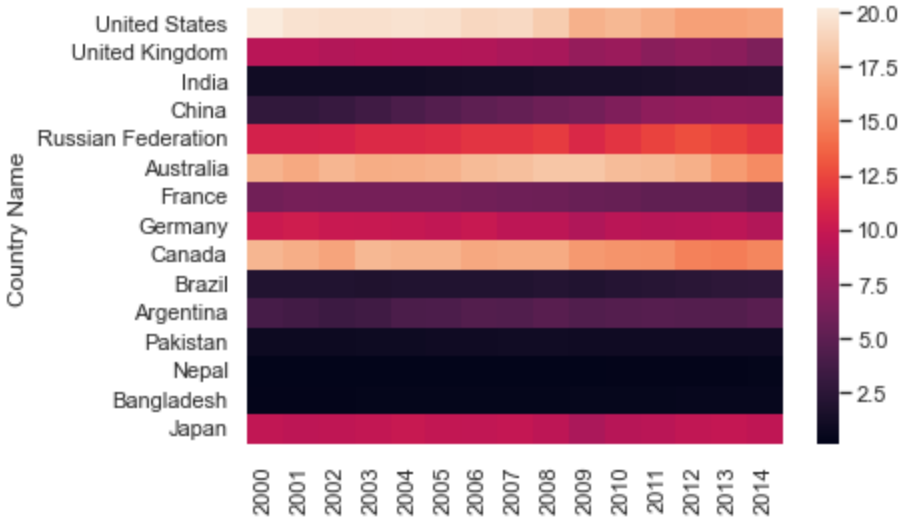
In [32]:
```
globalWarming_df=globalWarming_df.drop(columns=['Country Code','Indicator Name','Indicato
globalWarming_df.head(6)
```

Out[32]:

Country Name	2000	2001	2002	2003	2004	2005	2006	2007	
United States	20.178751	19.636505	19.613404	19.564105	19.658371	19.591885	19.094067	19.217898	1
United Kingdom	9.199549	9.233175	8.904123	9.053278	8.989140	8.982939	8.898710	8.617164	
India	0.979870	0.971698	0.967381	0.992392	1.025028	1.068563	1.121982	1.193210	
China	2.696862	2.742121	3.007083	3.524074	4.037991	4.523178	4.980314	5.334910	
Russian Federation	10.627121	10.669603	10.715901	11.090647	11.120627	11.253529	11.669122	11.672457	1
Australia	17.200610	16.733367	17.370452	16.901959	17.026515	17.169711	17.651398	17.865260	1

In [33]:
```
sns.heatmap(globalWarming_df)
```

Out[33]:
```
<AxesSubplot:ylabel='Country Name'>
```

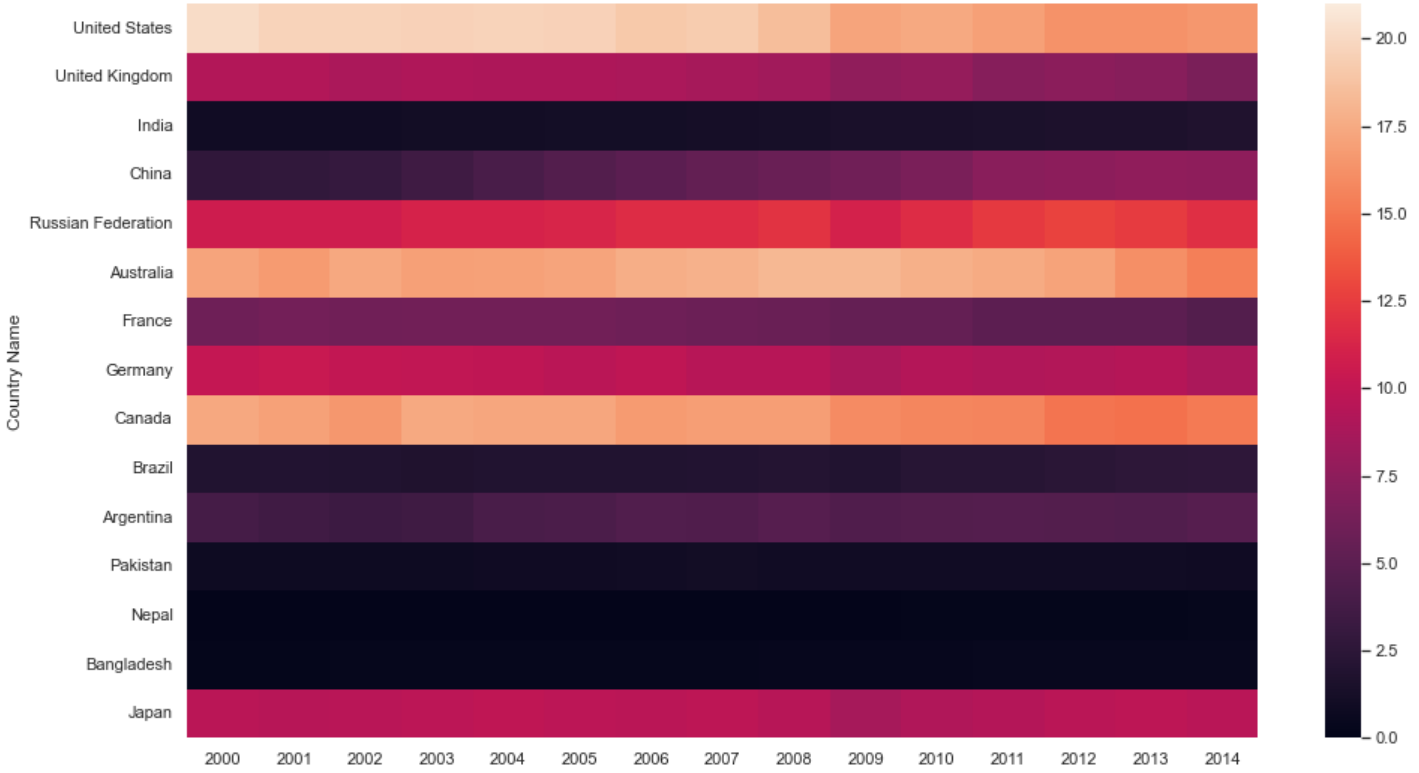


In [34]:
```
parameter set limit of colormap
```

```
plt.figure(figsize=(16,9))

sns.heatmap(globalWarming_df, vmin = 0, vmax = 21)
```

Out[34]:

```
<AxesSubplot:ylabel='Country Name'>
```

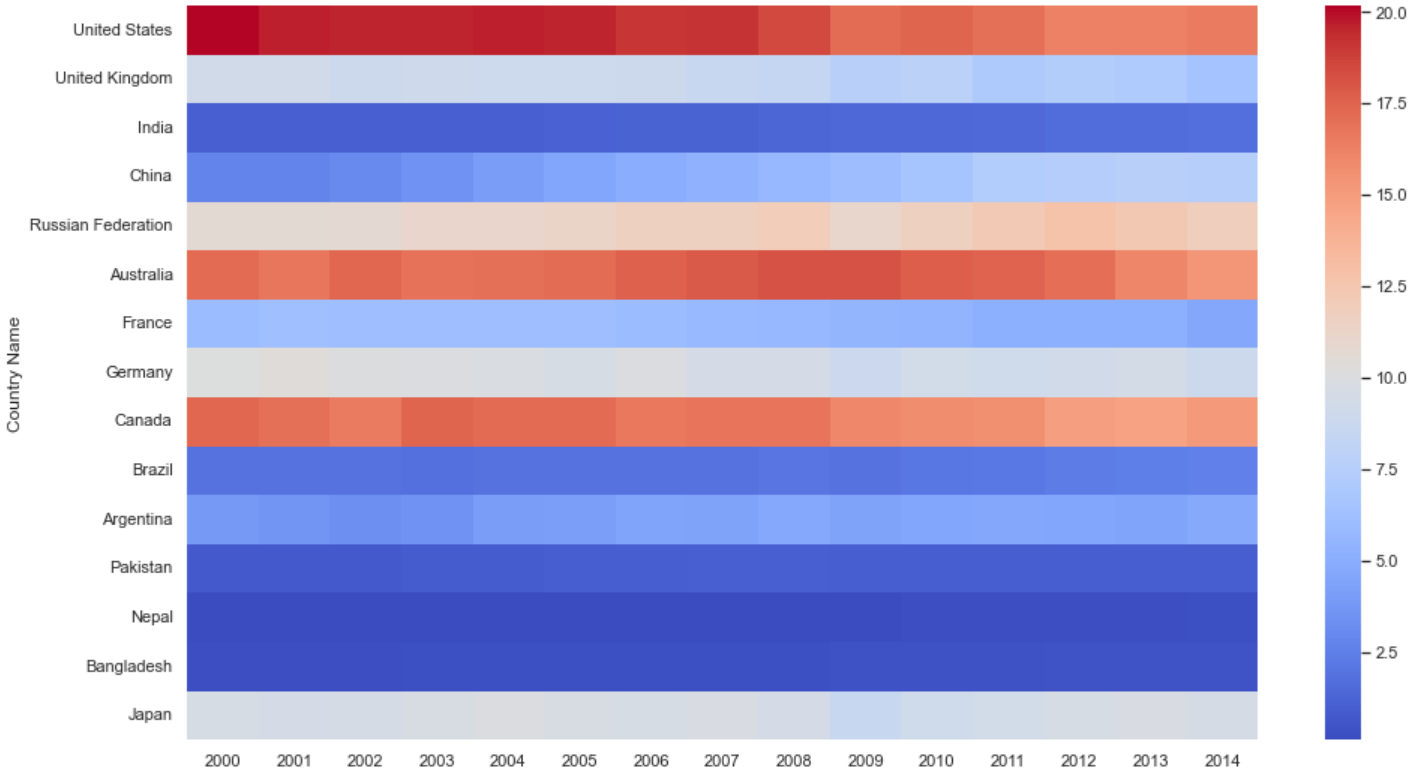


In [35]:

```
# change heatmap color using cmap

plt.figure(figsize=(16,9))

sns.heatmap(globalWarming_df, cmap="coolwarm")

#Possible values are: Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn, BuGn_r, BuPu,
#PiYG_r, PuBu, PuBuGn, PuBuGn_r, PuBu_r, PuOr, PuOr_r, PuRd, PuRd_r, Purples, Purples_r,
#YlOrBr, YlOrBr_r, YlOrRd, YlOrRd_r, afmhot, afmhot_r, autumn, autumn_r, binary, binary_r
#gist_heat, gist_heat_r, gist_ncar, gist_ncar_r, gist_rainbow, gist_rainbow_r, gist_stern
# mako_r, nipy_spectral, nipy_spectral_r, ocean, ocean_r, pink, pink_r, plasma, plasma_r,
#twilight, twilight_r, twilight_shifted, twilight_shifted_r, viridis, viridis_r, vlag, vl
```

Out[35]:

```
<AxesSubplot:ylabel='Country Name'>
```

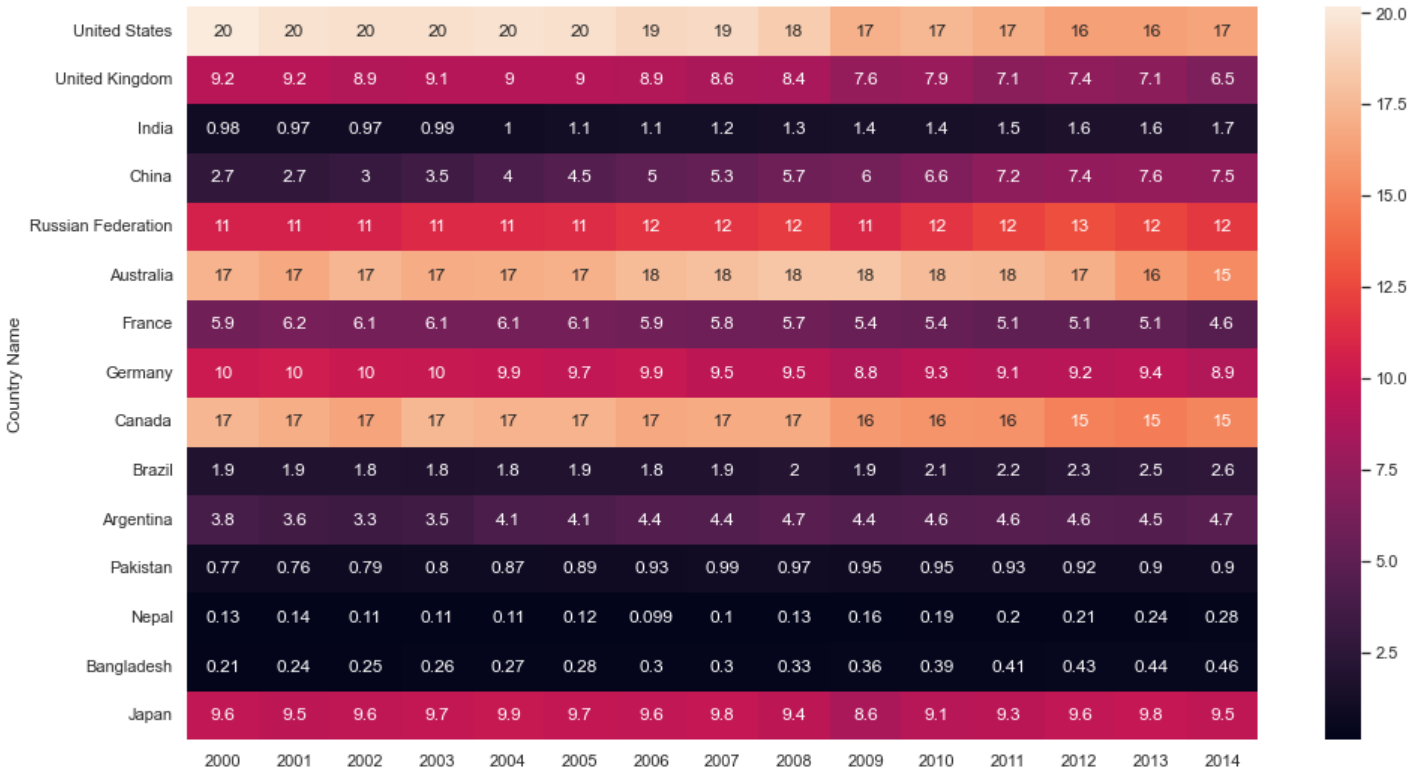


In [36]:

```
# annot (annotate) parameter to represent values

plt.figure(figsize=(16,9))

sns.heatmap(globalWarming_df, annot = True)
```

Out[36]:

```
<AxesSubplot:ylabel='Country Name'>
```



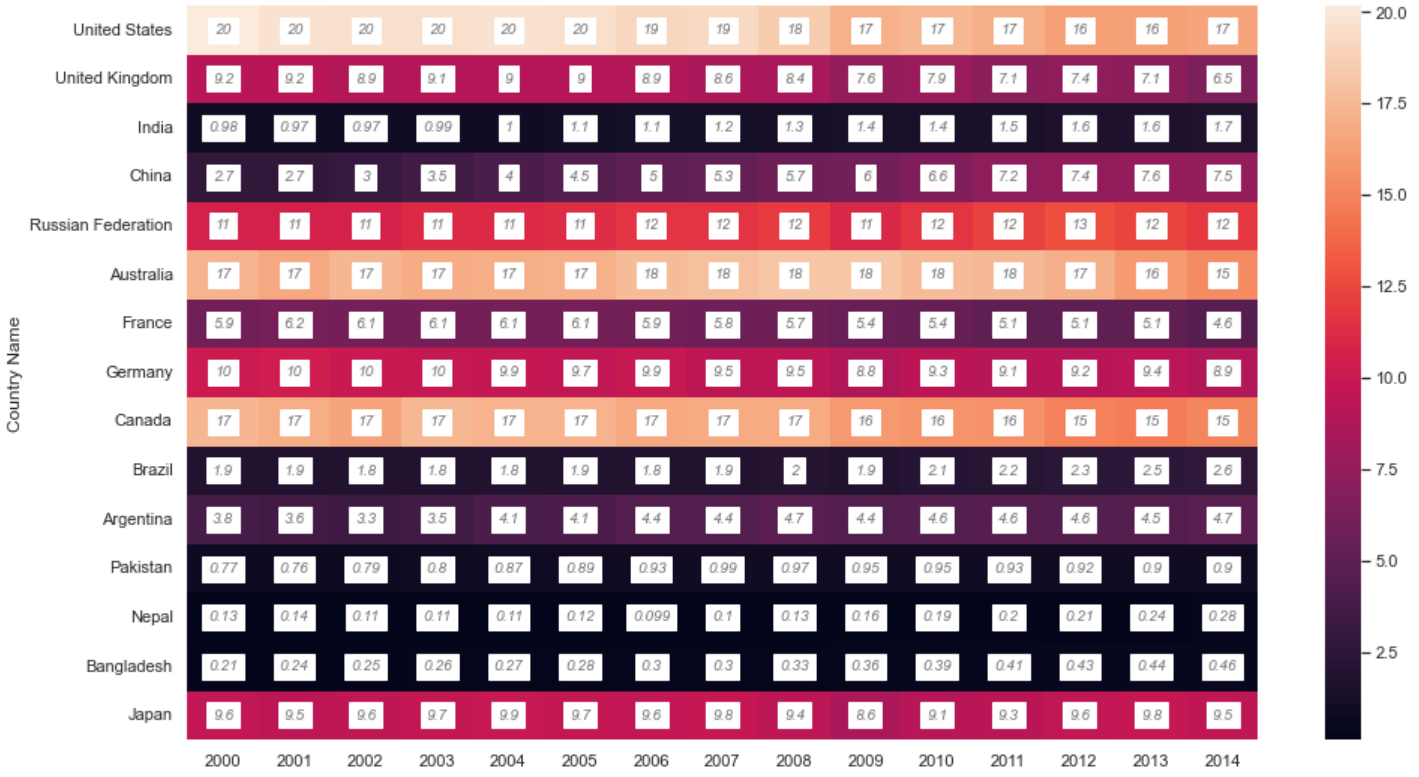
In [37]:

```
# annot_kws parameter

plt.figure(figsize=(16,9))

sns.heatmap(globalWarming_df, annot = True, annot_kws={'size':10,
```

```
            'fontstyle':'italic',
            'color':"k",
            'alpha':0.6,
            'rotation':"horizontal",
            'verticalalignment':'center',
            'backgroundcolor':'w'}

sns.heatmap(globalWarming_df, annot = True, annot_kws= annot_kws)
```

Out[37]:

```
<AxesSubplot:ylabel='Country Name'>
```

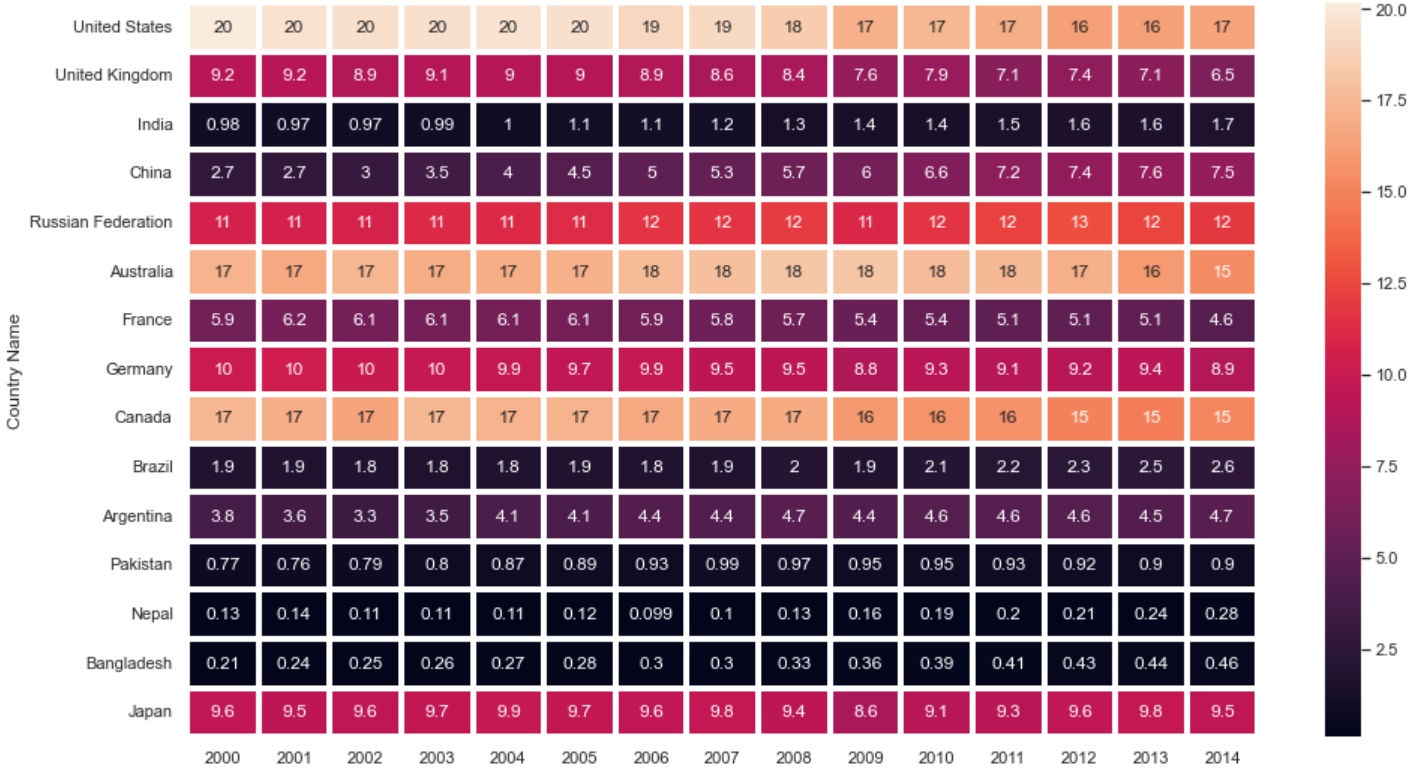


In [38]:

```
# linewidths parameter - divide each cell of heatmap

plt.figure(figsize=(16,9))

sns.heatmap(globalWarming_df, linewidths=4,annot=True)
```

Out[38]:

```
<AxesSubplot:ylabel='Country Name'>
```

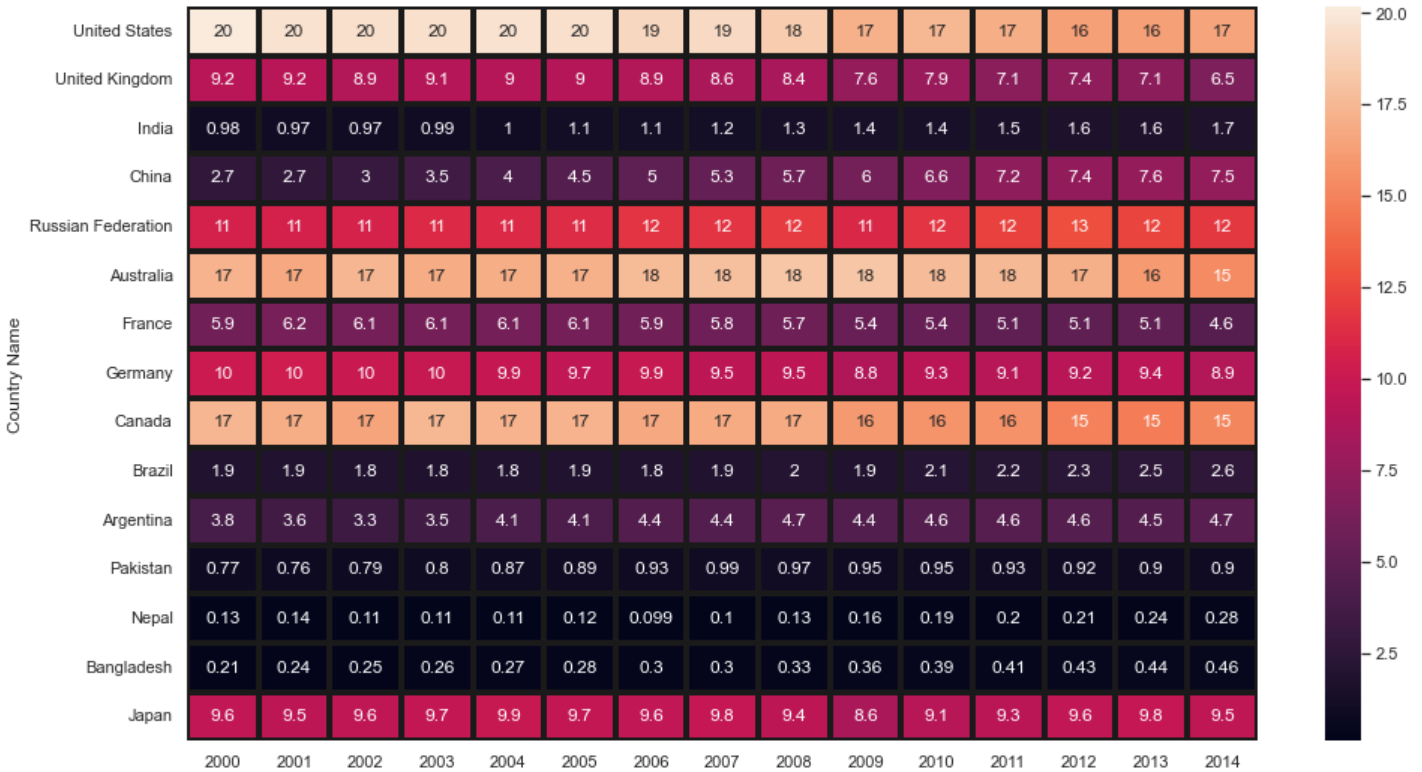


In [39]:

```
# linecolor parameter - change the color of heatmap line

plt.figure(figsize=(16,9))

sns.heatmap(globalWarming_df, linewidths=4, linecolor="k",annot=True)
```

Out[39]:

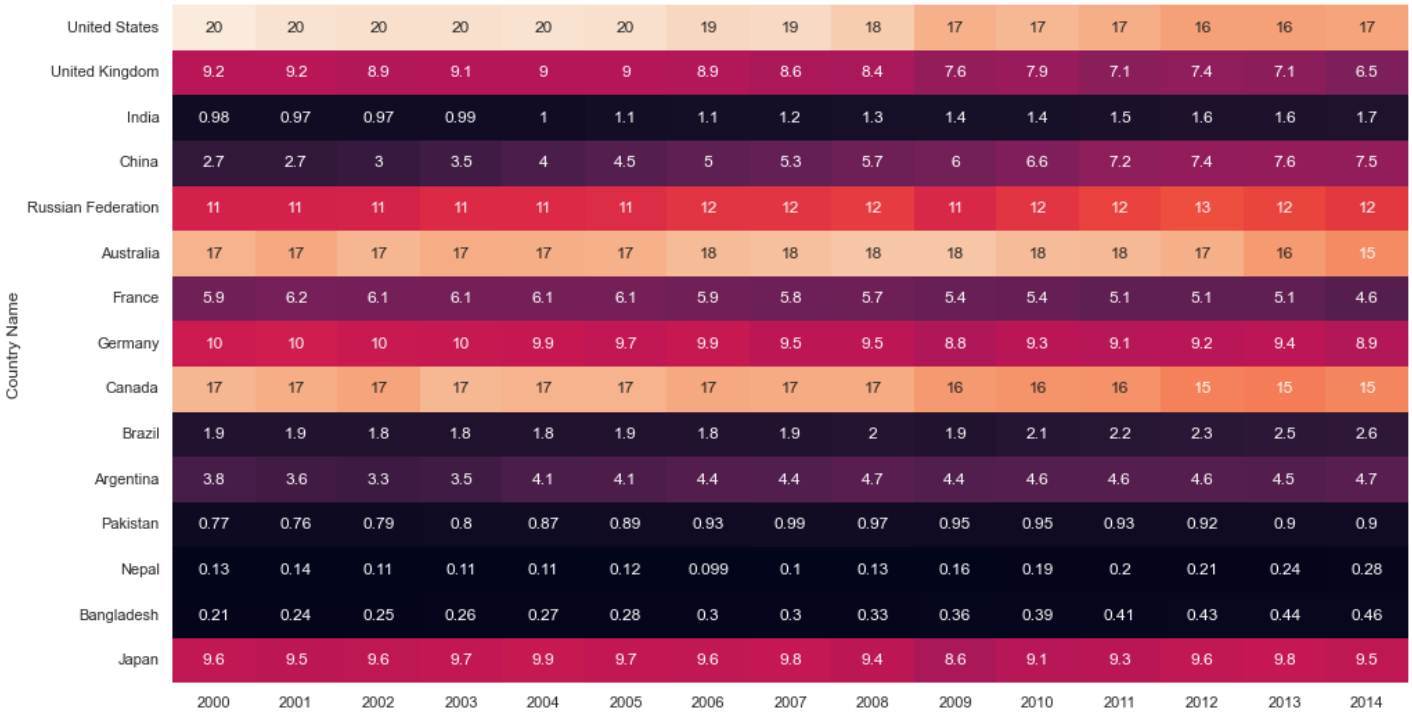
```
<AxesSubplot:ylabel='Country Name'>
```



In [40]:

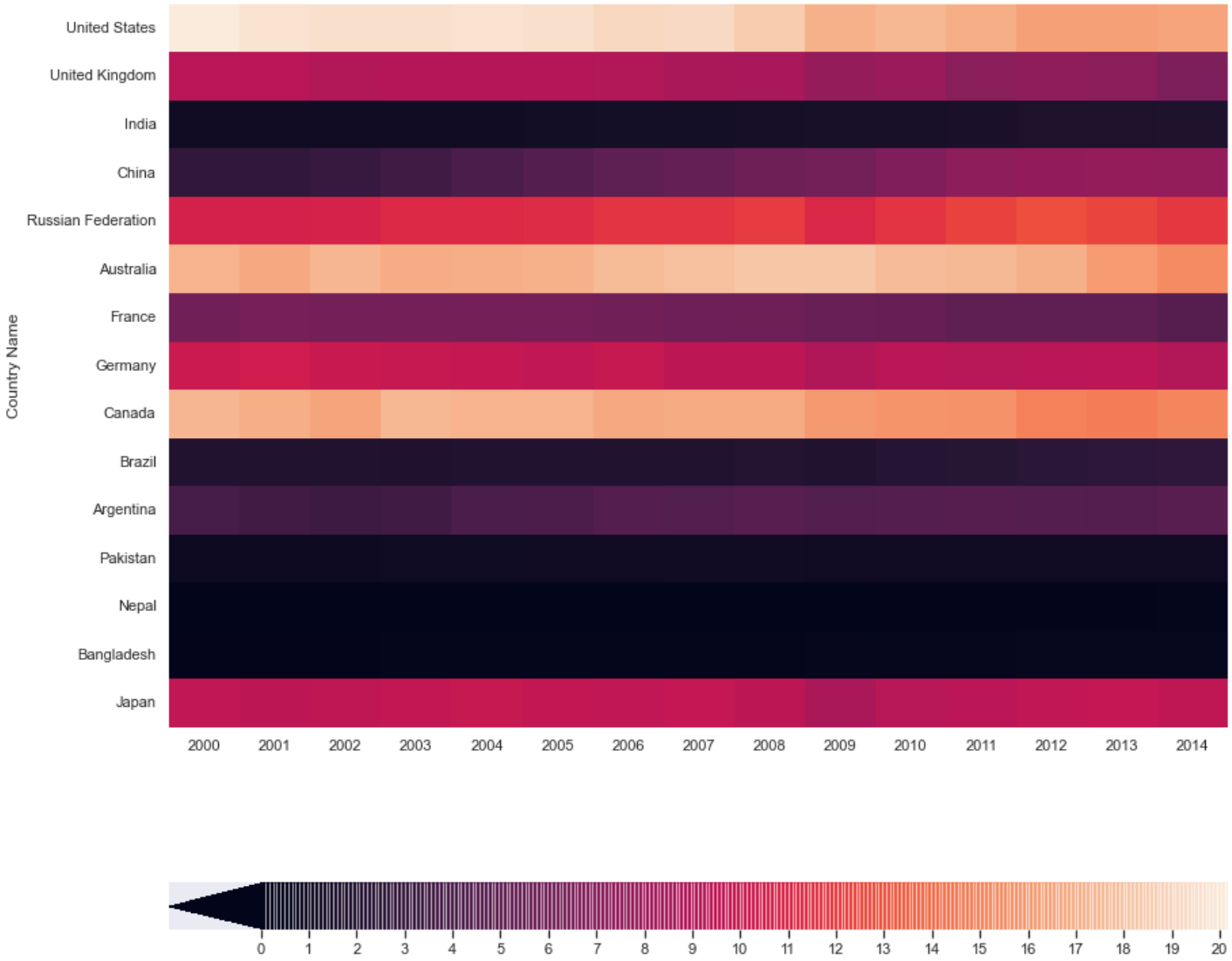
```
# hide color bar with cbar parameter

plt.figure(figsize=(16,9))

                balWarming_df, cbar = False,annot=True)
```

Out[40]: <AxesSubplot:ylabel='Country Name'>



In [41]:
```
# change style and format of color bar with cbar_kws parameter

plt.figure(figsize=(14,14))

cbar_kws = {"orientation":"horizontal",
            "shrink":1,
            'extend':'min',
            'extendfrac':0.1,
            "ticks":np.arange(0,22),
            "drawedges":True,
           }

sns.heatmap(globalWarming_df, cbar_kws=cbar_kws)
#shrink: To change the size of the color bar
#extend: To change the end of the color bar like pointed or not.
#If you want pointed color bar both side then passes value ‘both’, for left ‘min’ , right
#extendfrac: To adjust the extension of the color bar.
#The ‘auto’ value adjust pointer automatically, ‘False’ value for no pointer and float va
#drawedges: To draw lines (edges) on the color bar.
```

Out[41]: <AxesSubplot:ylabel='Country Name'>

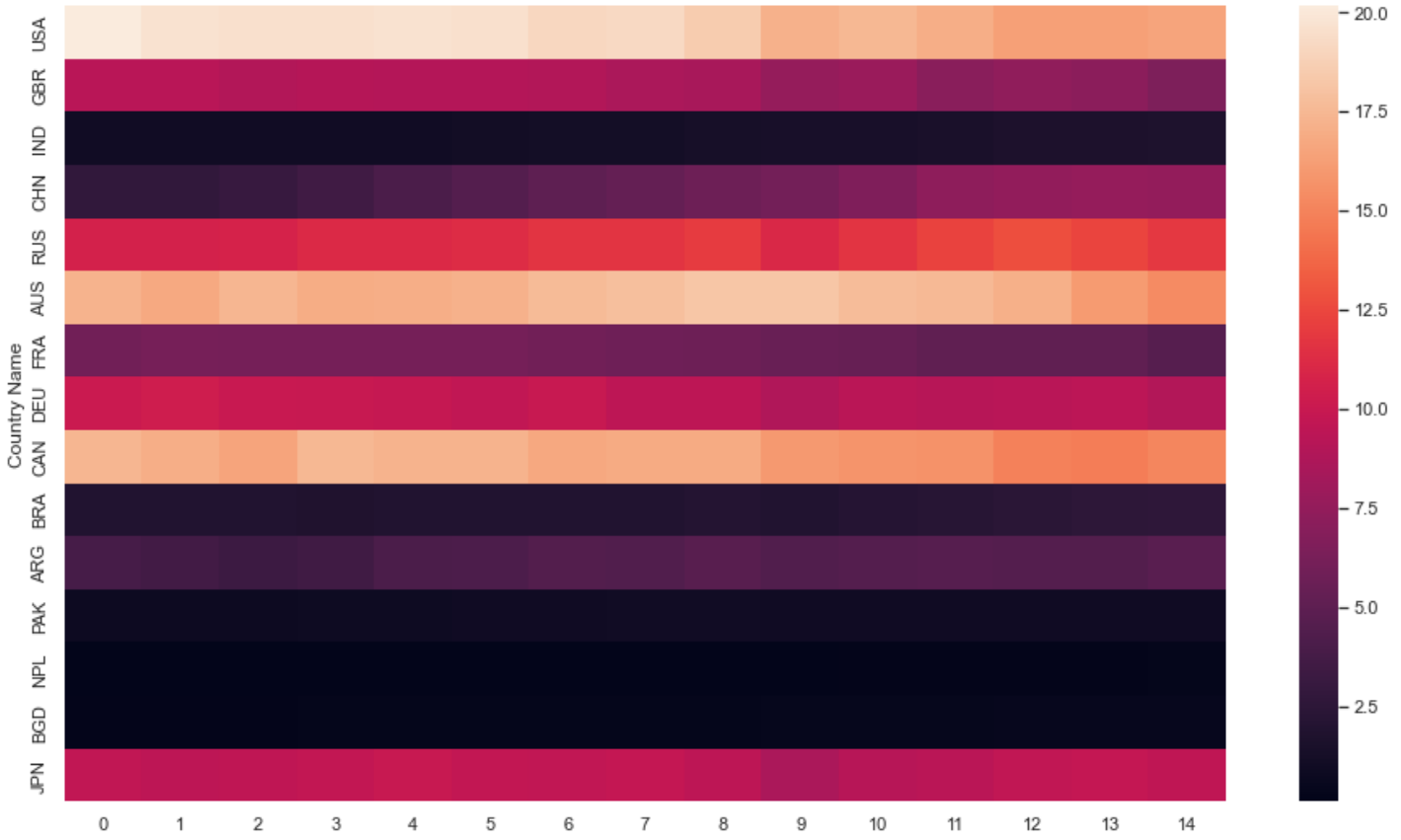


In [42]:

```
# change y-axis labels using yticklabels parameter

plt.figure(figsize=(16,9))

country_code = ['USA', 'GBR', 'IND', 'CHN', 'RUS', 'AUS', 'FRA', 'DEU', 'CAN', 'BRA', 'AR

sns.heatmap(globalWarming_df, yticklabels = country_code, xticklabels = np.arange(0,15))
```

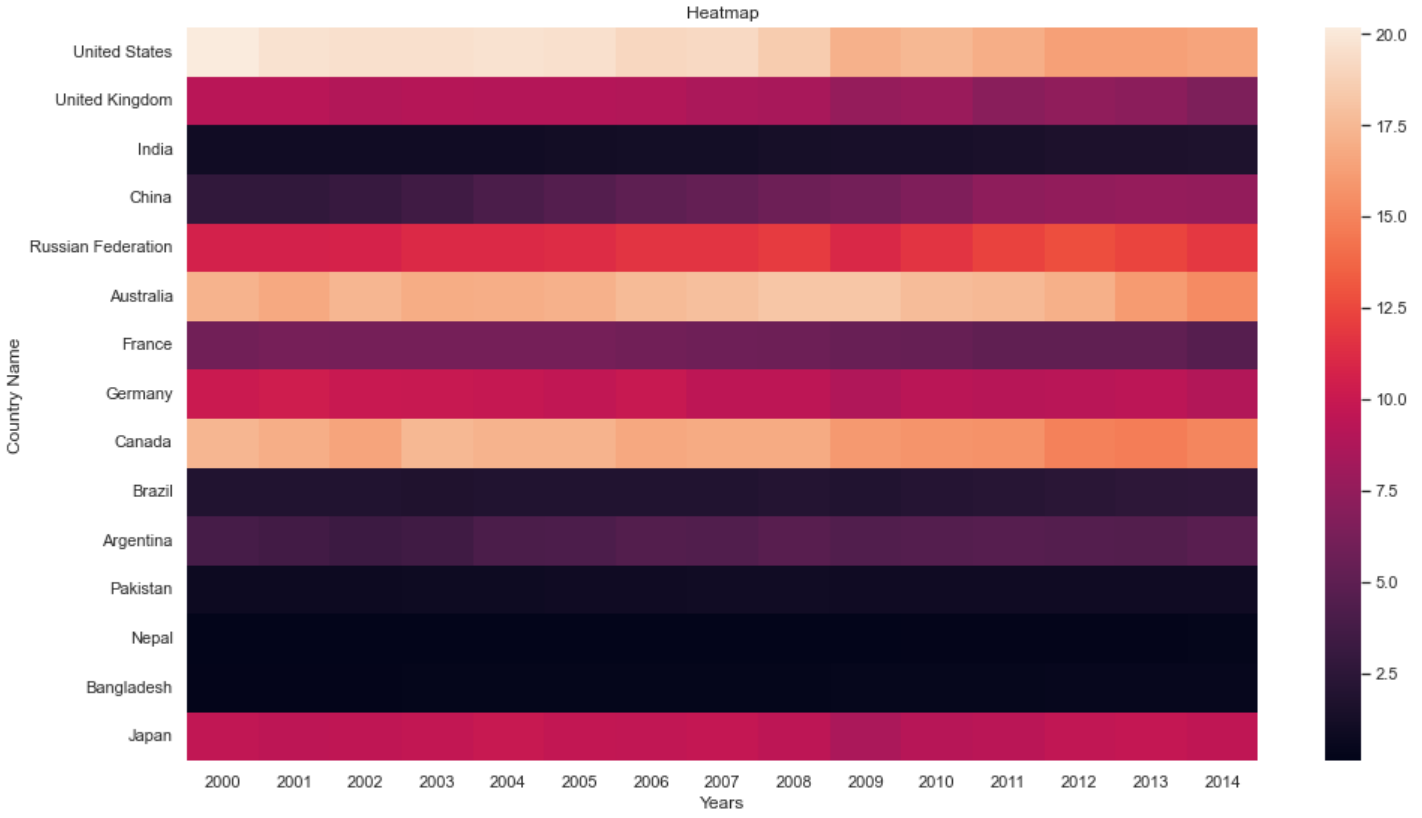
Out[42]:

```
<AxesSubplot:ylabel='Country Name'>
```



In [43]:

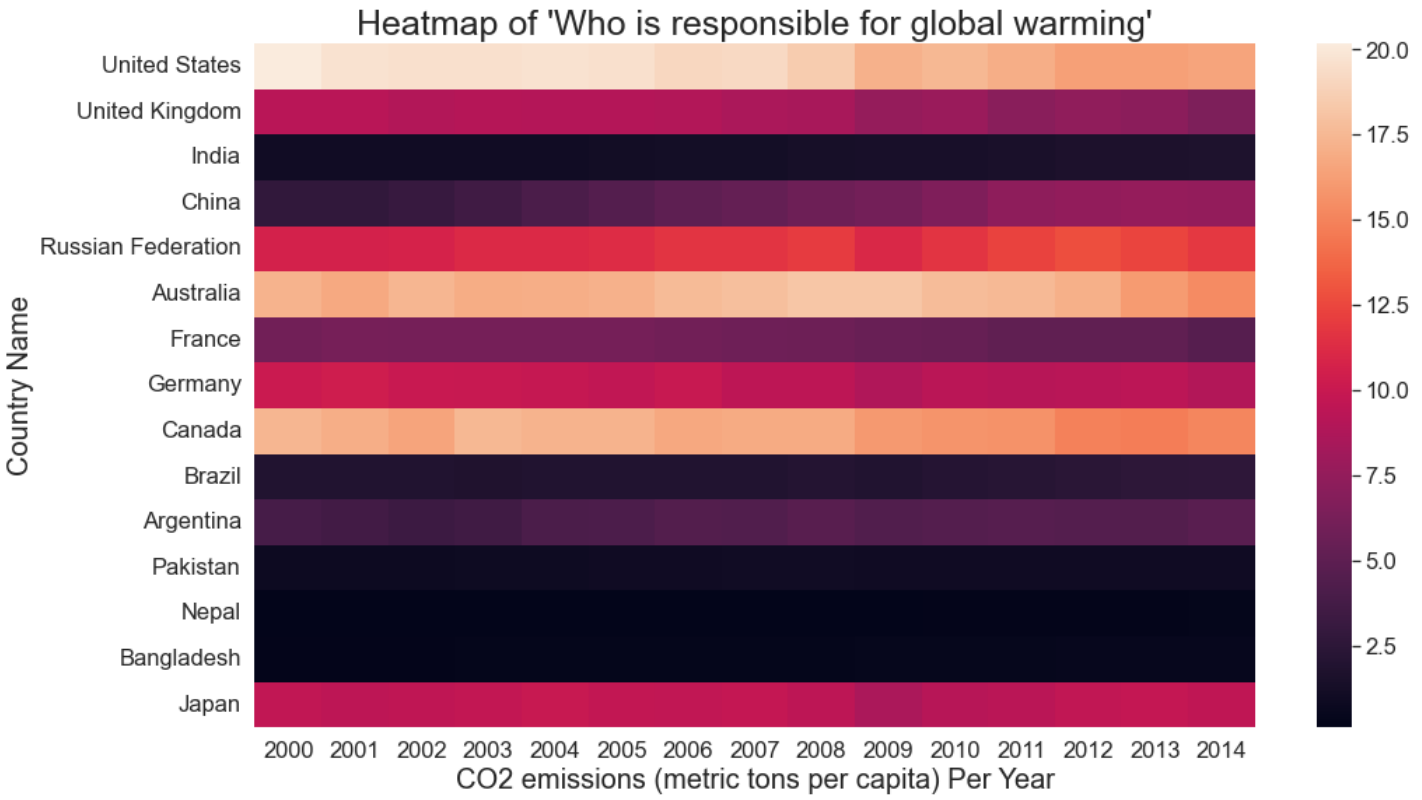
```
# set seaborn heatmap title, x-axis, y-axis label and font size

plt.figure(figsize=(16,9))

ax = sns.heatmap(globalWarming_df)

ax.set(title="Heatmap",
       xlabel="Years",
       ylabel="Country Name",)

sns.set(font_scale=1.5) # set fontsize 2
# Axes parameter help to set multiple things like heatmap title, x-axis, y-axis labels
```



Example 1:

In [44]:

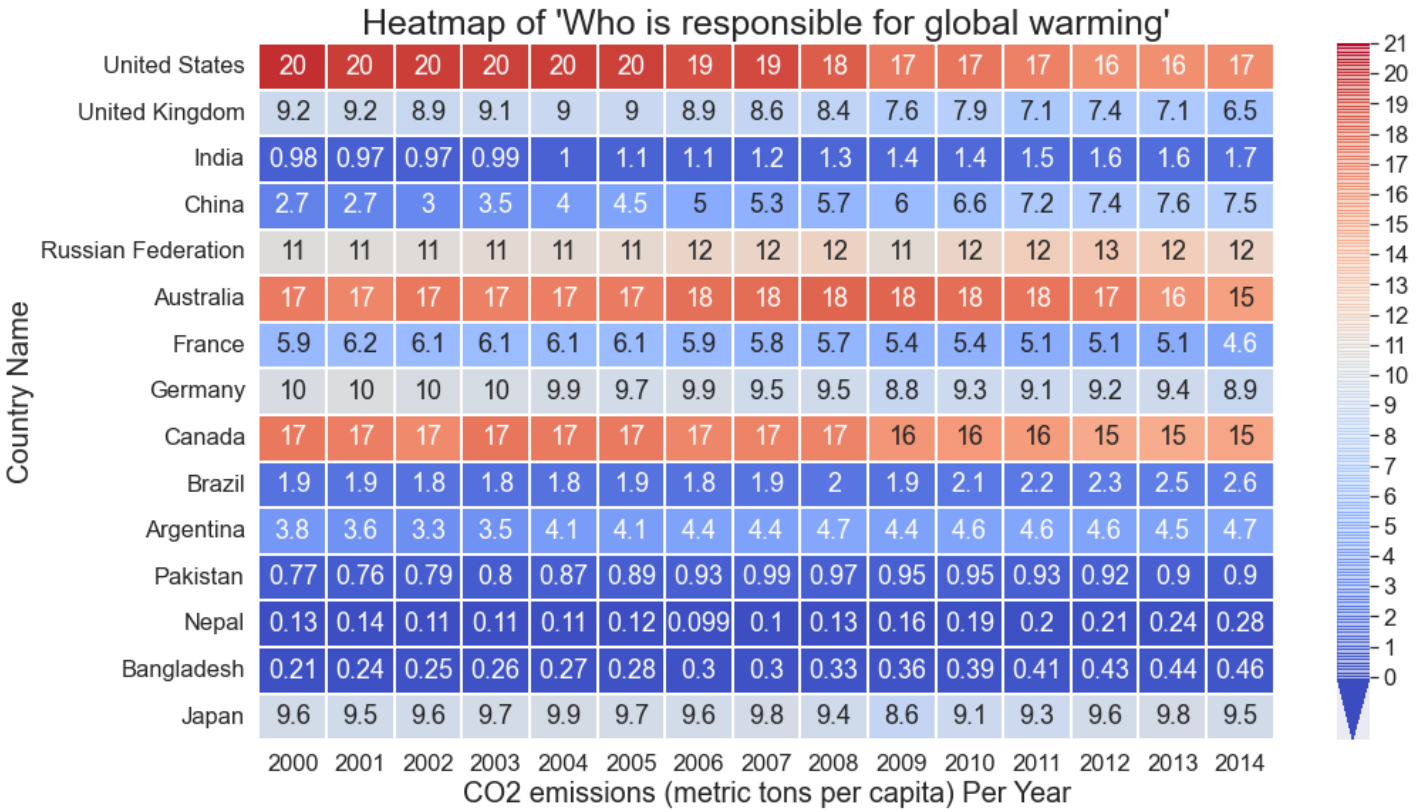
```
# import libraries
import seaborn as sns # for data visualization
import matplotlib.pyplot as plt # for data visualization
import pandas as pd # for data analysis

# load dataset and create DataFrame ready to create heatmap
globalWarming_df = pd.read_csv("E:\\New folder\\Who_is_responsible_for_global_warming.csv
globalWarming_df = globalWarming_df.drop(columns=['Country Code', 'Indicator Name', 'Indi

# set heatmap size
plt.figure(figsize= (16,9))

# create heatmap seaborn
sns.heatmap(globalWarming_df)

plt.title("Heatmap of 'Who is responsible for global warming'", fontsize = 25)
plt.xlabel("CO2 emissions (metric tons per capita) Per Year", fontsize = 20)
plt.ylabel("Country Name", fontsize = 20)
plt.show()
```



Example 2:

In [45]:

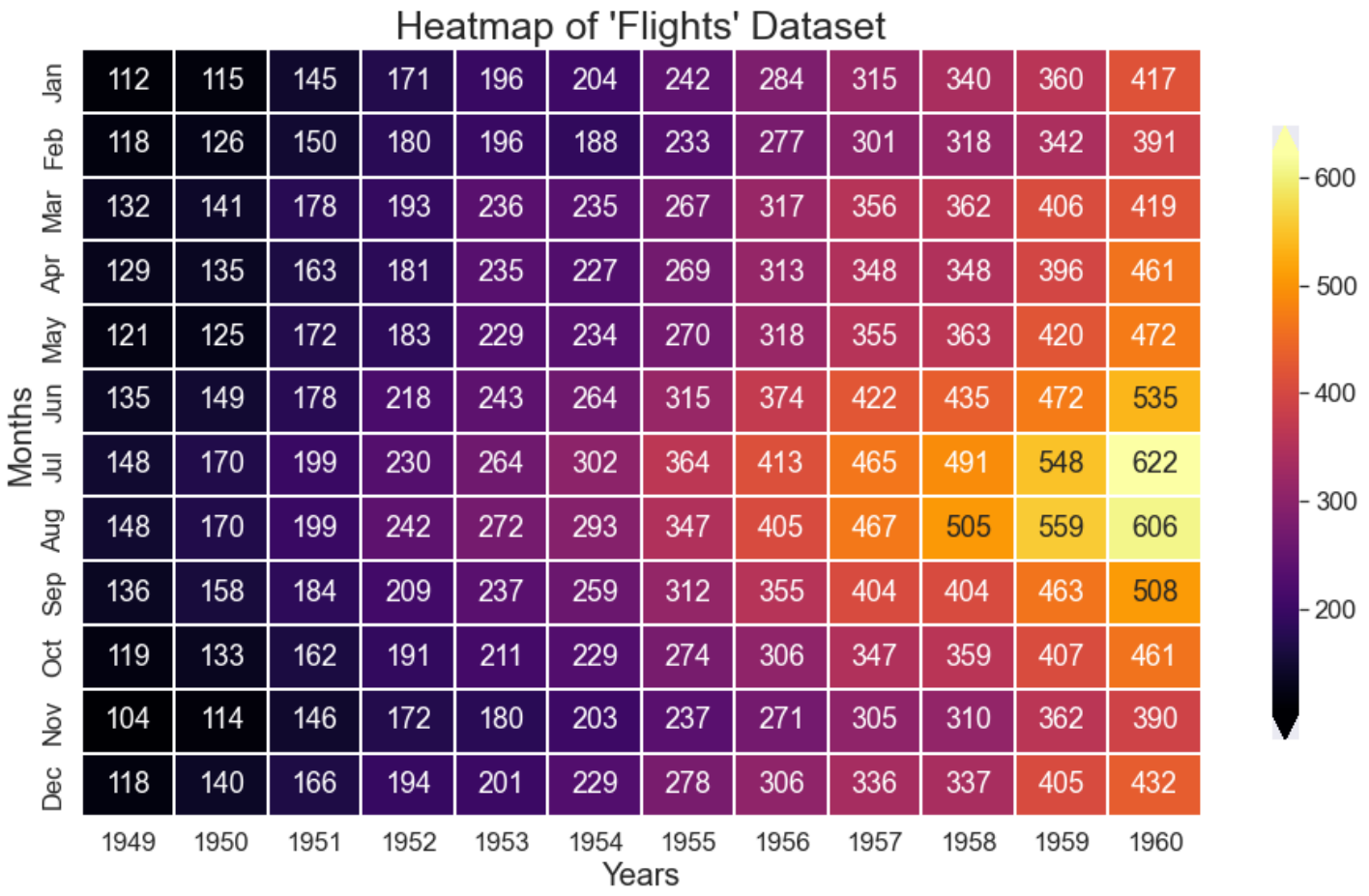
```
# import libraries
import seaborn as sns # for data visualization
import matplotlib.pyplot as plt # for data visualization
import pandas as pd # for data analysis

# load dataset and create DataFrame ready to create heatmap
globalWarming_df = pd.read_csv("E:\\New folder\\Who_is_responsible_for_global_warming.csv
globalWarming_df = globalWarming_df.drop(columns=['Country Code', 'Indicator Name', 'Indi

# set heatmap size
plt.figure(figsize= (16,9))

# create heatmap seaborn

cbar_kws = {"orientation":"vertical",
            "shrink":1,
            'extend':'min',
            'extendfrac':0.1,
            "ticks":np.arange(0,22),
            "drawedges":True,
           } # color bar keyword arguments

sns.heatmap(globalWarming_df, vmin = 0, vmax = 21, cmap="coolwarm", annot = True, linewid

plt.title("Heatmap of 'Who is responsible for global warming'", fontsize = 25)
plt.xlabel("CO2 emissions (metric tons per capita) Per Year", fontsize = 20)
plt.ylabel("Country Name", fontsize = 20)
plt.show()
```



Example 3:

In [46]:

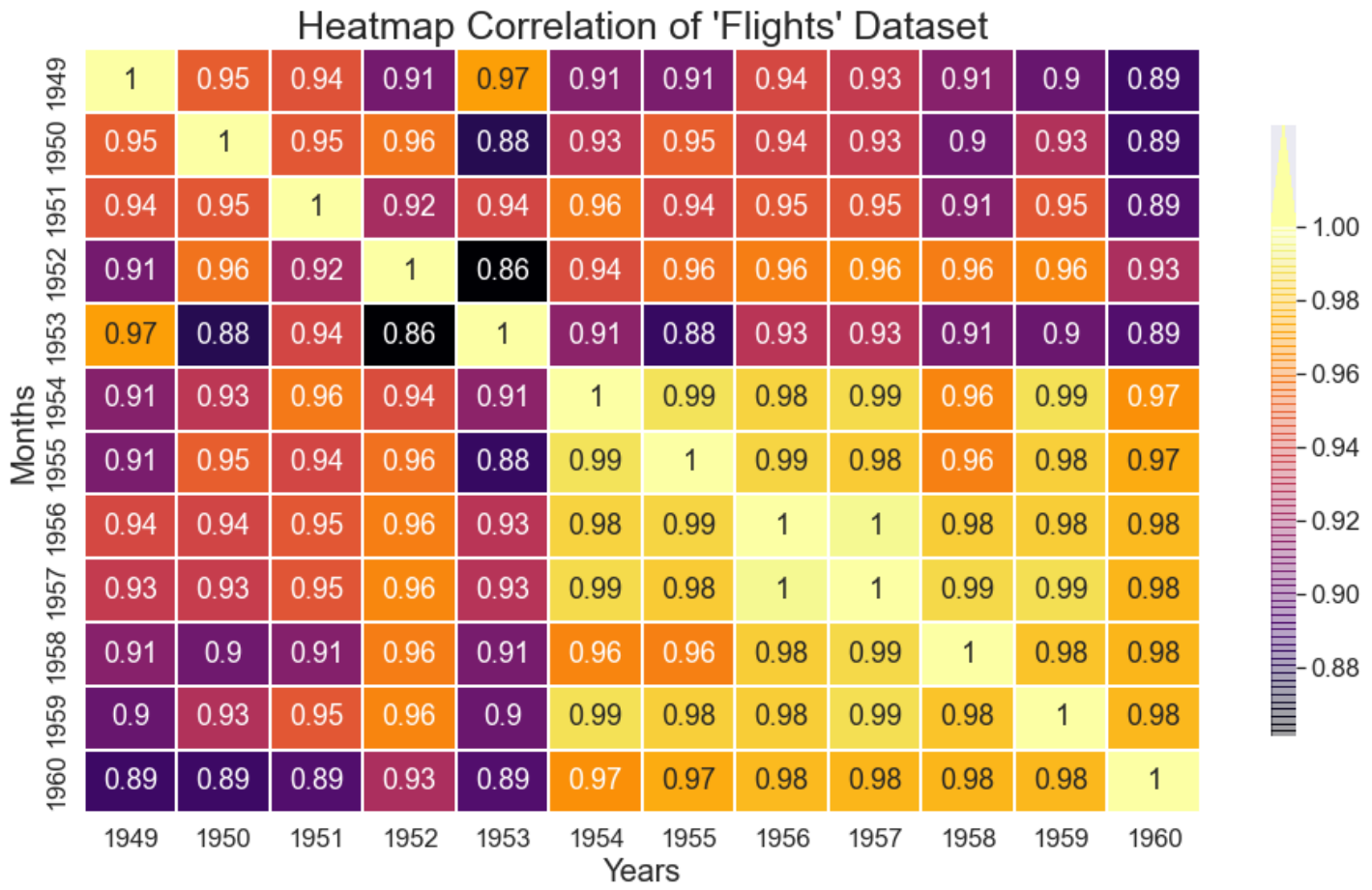
```
# import libraries
import seaborn as sns # for data visualization
import matplotlib.pyplot as plt # for data visualization
import pandas as pd # for data analysis

# load dataset and create DataFrame ready to create heatmap
flights = sns.load_dataset("flights")
flights_df = flights.pivot("month", "year", "passengers")

# set heatmap size
plt.figure(figsize= (16,9))

# create heatmap seaborn

cbar_kws = {"shrink":.8,
            'extend':'both'}

sns.heatmap(flights_df, cmap="inferno", annot = True, fmt = 'd', linewidth = 2, cbar_kws=

plt.title("Heatmap of 'Flights' Dataset", fontsize = 25)
plt.xlabel("Years", fontsize = 20)
plt.ylabel("Months", fontsize = 20)
plt.show()
```



Example 4:

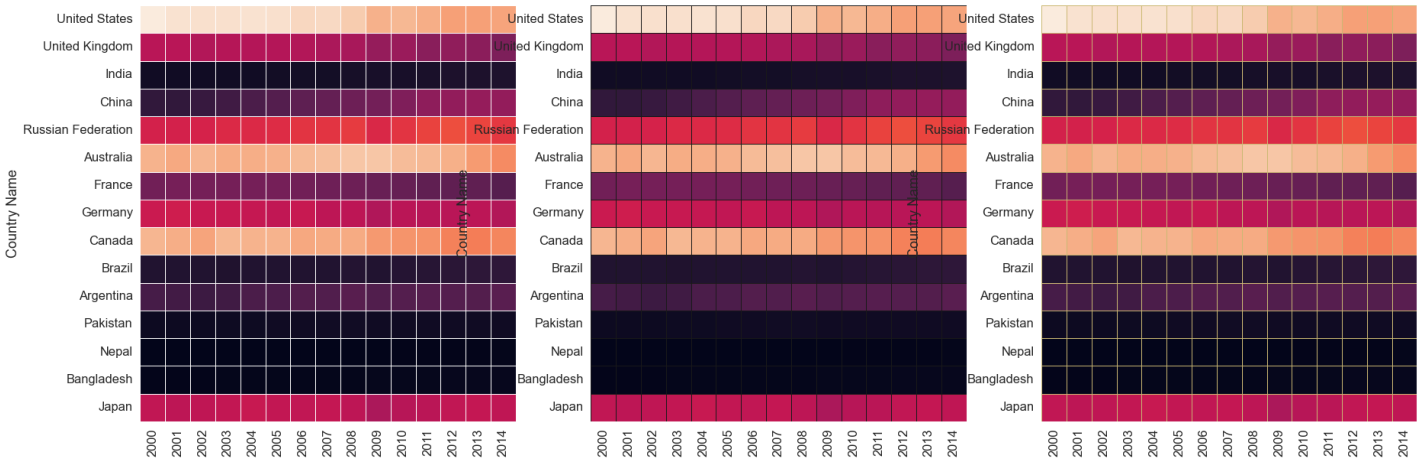
In [47]:

```
# load dataset and create DataFrame ready to create heatmap
flights = sns.load_dataset("flights")
flights_df = flights.pivot("month", "year", "passengers")

# set heatmap size
plt.figure(figsize= (16,9))

# create heatmap seaborn
# When you want to find what's the relationship between multiple features and which featu
#Machine Learning model building. Then take correlation of that dataset and visualize by

cbar_kws = {"shrink":.8,
            'extend':'max',
            'extendfrac':.2,
            "drawedges":True}

sns.heatmap(flights_df.corr(), cmap="inferno", annot = True, linewidth = 2, cbar_kws=cbar_

plt.title("Heatmap Correlation of 'Flights' Dataset", fontsize = 25)
plt.xlabel("Years", fontsize = 20)
plt.ylabel("Months", fontsize = 20)
plt.show()
```



In [48]:

```
# multiple heatmaps using subplots

plt.figure(figsize=(30,10))

plt.subplot(1,3,1) # first heatmap
sns.heatmap(globalWarming_df,  cbar=False, linecolor="w", linewidths=1)

plt.subplot(1,3,2) # second heatmap
sns.heatmap(globalWarming_df,  cbar=False, linecolor="k", linewidths=1)

plt.subplot(1,3,3) # third heatmap
sns.heatmap(globalWarming_df,  cbar=False, linecolor="y", linewidths=1)

plt.show()
```



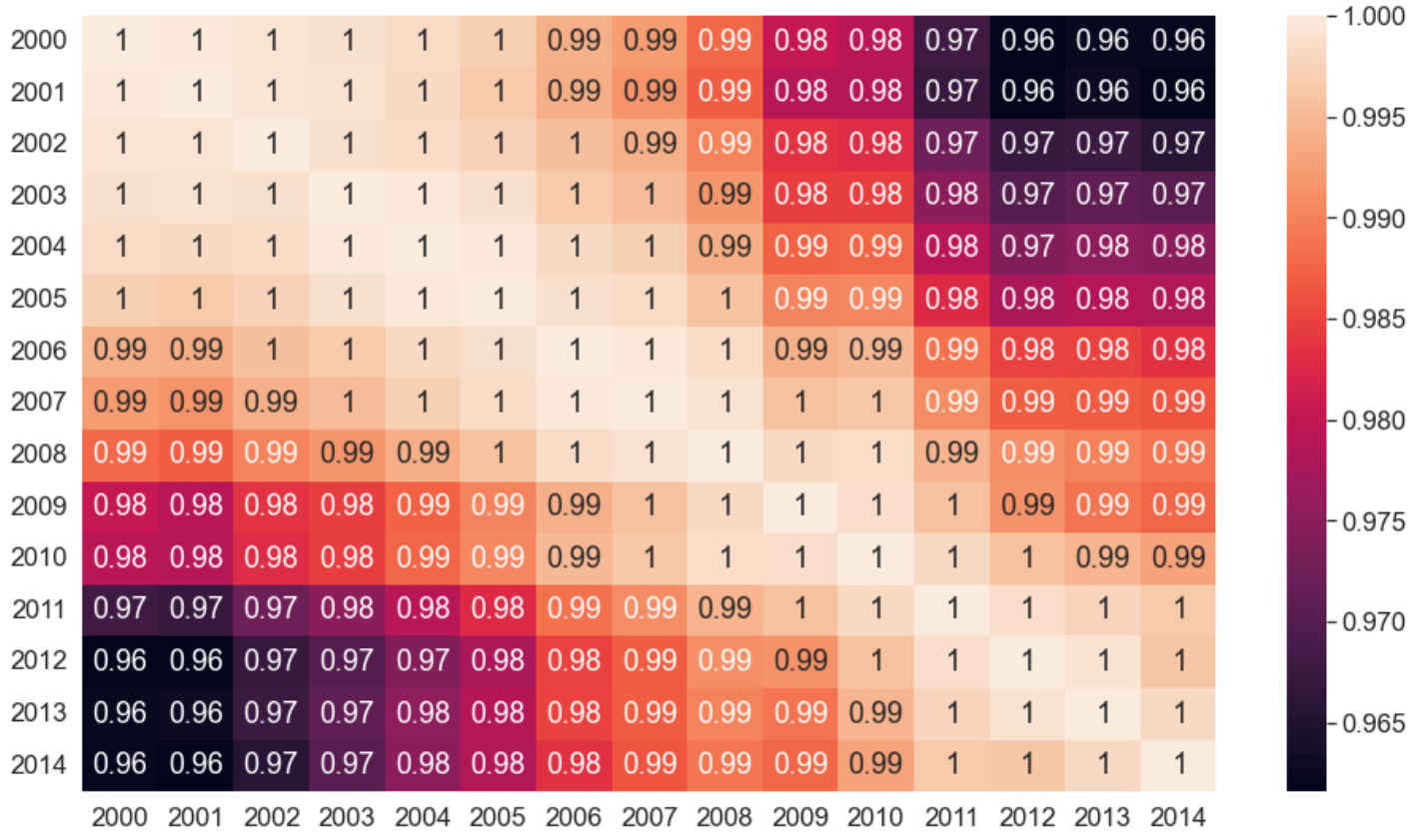
In [49]:

```
# sns heatmap correlation
#Corelation is a statiscal measure of relationship b/w two variable(X&Y).
#Range from -1 to 1
```

```
#-1:Negative(X->Increases,Y->Decreases)
#0:No effect
#1:Positive(X->Increases,Y->Increases)
plt.figure(figsize=(16,9))

sns.heatmap(globalWarming_df.corr(), annot = True)
```

Out[49]:

```
<AxesSubplot:>
```



In [50]:

```
from sklearn.datasets import load_breast_cancer
cancer_dataset = load_breast_cancer()
cancer_dataset
```

Out[50]:

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
```

```
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n---
----------------------------------------\n\n**Data Set Characteristics:**\n\n    :Number
of Instances: 569\n\n    :Number of Attributes: 30 numeric, predictive attributes and the
class\n\n    :Attribute Information:\n        - radius (mean of distances from center to p
oints on the perimeter)\n        - texture (standard deviation of gray-scale values)\n
- perimeter\n        - area\n        - smoothness (local variation in radius lengths)\n
- compactness (perimeter^2 / area - 1.0)\n        - concavity (severity of concave portion
s of the contour)\n        - concave points (number of concave portions of the contour)\n
- symmetry\n        - fractal dimension ("coastline approximation" - 1)\n\n        The mea
n, standard error, and "worst" or largest (mean of the three\n        worst/largest value
s) of these features were computed for each image,\n        resulting in 30 features.  For
instance, field 0 is Mean Radius, field\n        10 is Radius SE, field 20 is Worst Radiu
s.\n\n        - class:\n                - WDBC-Malignant\n                - WDBC-Benign\n
\n    :Summary Statistics:\n\n    ===================================== ====== ======\n
Min    Max\n    ===================================== ====== ======\n    radius (mean):
6.981  28.11\n    texture (mean):                         9.71   39.28\n    perimeter (mea
n):                  43.79  188.5\n    area (mean):                            143.5  250
1.0\n    smoothness (mean):                   0.053  0.163\n    compactness (mean):
0.019  0.345\n    concavity (mean):                       0.0    0.427\n    concave points
(mean):               0.0    0.201\n    symmetry (mean):                        0.106  0.30
4\n    fractal dimension (mean):             0.05   0.097\n    radius (standard error):
0.112  2.873\n    texture (standard error):             0.36   4.885\n    perimeter (stand
ard error):          0.757  21.98\n    area (standard error):                  6.802  542.2
\n    smoothness (standard error):         0.002  0.031\n    compactness (standard erro
r):         0.002  0.135\n    concavity (standard error):           0.0    0.396\n    conc
ave points (standard error):      0.0    0.053\n    symmetry (standard error):
0.008  0.079\n    fractal dimension (standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                        12.02  49.54\n    perimeter (wors
t):                  50.41  251.2\n    area (worst):                           185.2  425
4.0\n    smoothness (worst):                  0.071  0.223\n    compactness (worst):
0.027  1.058\n    concavity (worst):                      0.0    1.252\n    concave points
(worst):              0.0    0.291\n    symmetry (worst):                       0.156  0.66
4\n    fractal dimension (worst):            0.055  0.208\n    =====================
========== ====== ======\n\n    :Missing Attribute Values: None\n\n    :Class Distributio
n: 212 - Malignant, 357 - Benign\n\n    :Creator:  Dr. William H. Wolberg, W. Nick Street,
Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: November, 1995\n\nThis is a c
opy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://goo.gl/U2Uwz2\n\nFea
tures are computed from a digitized image of a fine needle\naspirate (FNA) of a breast mas
s.  They describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating
plane described above was obtained using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett,
"Decision Tree\nConstruction Via Linear Programming." Proceedings of the 4th\nMidwest Arti
ficial Intelligence and Cognitive Science Society,\npp. 97-101, 1992], a classification me
thod which uses linear\nprogramming to construct a decision tree.  Relevant features\nwere
selected using an exhaustive search in the space of 1-4\nfeatures and 1-3 separating plane
s.\n\nThe actual linear program used to obtain the separating plane\nin the 3-dimensional
space is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgram
```

```
ming Discrimination of Two Linearly Inseparable Sets",\nOptimization Methods and Software
1, 1992, 23-34].\n\nThis database is also available through the UW CS ftp server:\n\nftp f
tp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topic:: References\n\n
- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction \n     for br
east tumor diagnosis. IS&T/SPIE 1993 International Symposium on \n     Electronic Imaging:
Science and Technology, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L. M
angasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n     prognosis via
linear programming. Operations Research, 43(4), pages 570-577, \n     July-August 1995.\n
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques\n     to di
agnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) \n     163-17
1.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\bre
ast_cancer.csv'}
```

In [51]:

```
cancer_df = pd.DataFrame(np.c_[cancer_dataset['data'],cancer_dataset['target']],#c_ to co
             columns = np.append(cancer_dataset['feature_names'], ['target']))
cancer_df.head(6)
```

Out[51]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	di
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.1578	0.08089	0.2087	

6 rows × 31 columns

In [52]:

```
#https://indianaiproduction.com/seaborn-pairplot/
#To see more on pairplots use it cuz it takes much time to load
```