i.relational plots:this plot is used to understand the relation between two variables

ii.categorical plots: this plot deals with categorical variables and how they are visualized.

iii.distribution plots:this plot is used for examining univariate and bivariate distribution.

iv.matrix plots: a matrix plot is an array ofscatterplots.

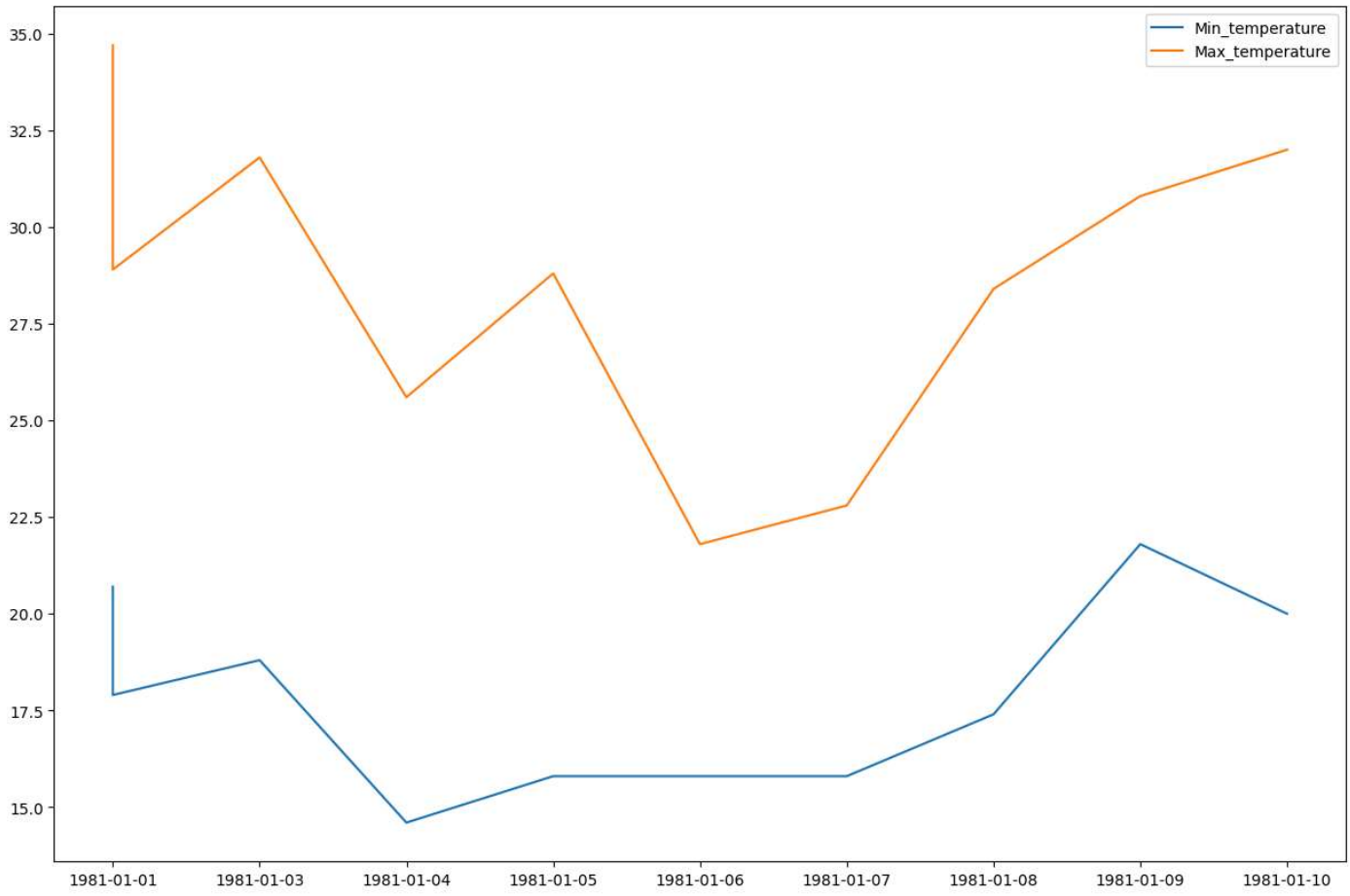
v.regression plots:regressionm plotrs in searborn are primarily intende to add a visual guide that helps to emphasize pattern in a dataset during exploratory data analysis.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
```

```
import seaborn as sns
%matplotlib inline
```

```
#Simple plotting with seaborn
dates = ['1981-01-01','1981-01-01','1981-01-03','1981-01-04','1981-01-05','1981-01-06','1981-01-07','1981-01-08','1981-01-09','1981-01-10']
min_temperature=[20.7,17.9,18.8,14.6,15.8,15.8,15.8,17.4,21.8,20.0]
max_temperature=[34.7,28.9,31.8,25.6,28.8,21.8,22.8,28.4,30.8,32.0]
fig,axes=plt.subplots(nrows=1,ncols=1,figsize=(15,10))
axes.plot(dates, min_temperature, label='Min_temperature')
axes.plot(dates, max_temperature, label='Max_temperature')
axes.legend()
```

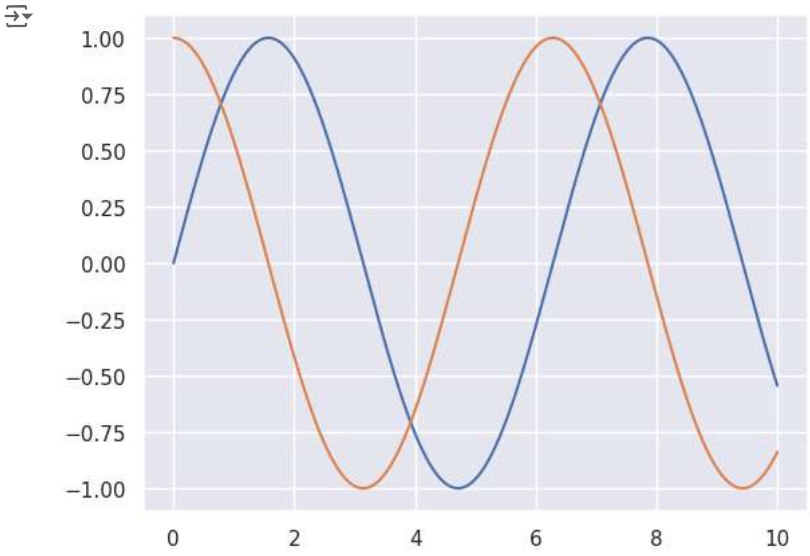
```
<matplotlib.legend.Legend at 0x7e60798767d0>
```



```
sns.set()
```

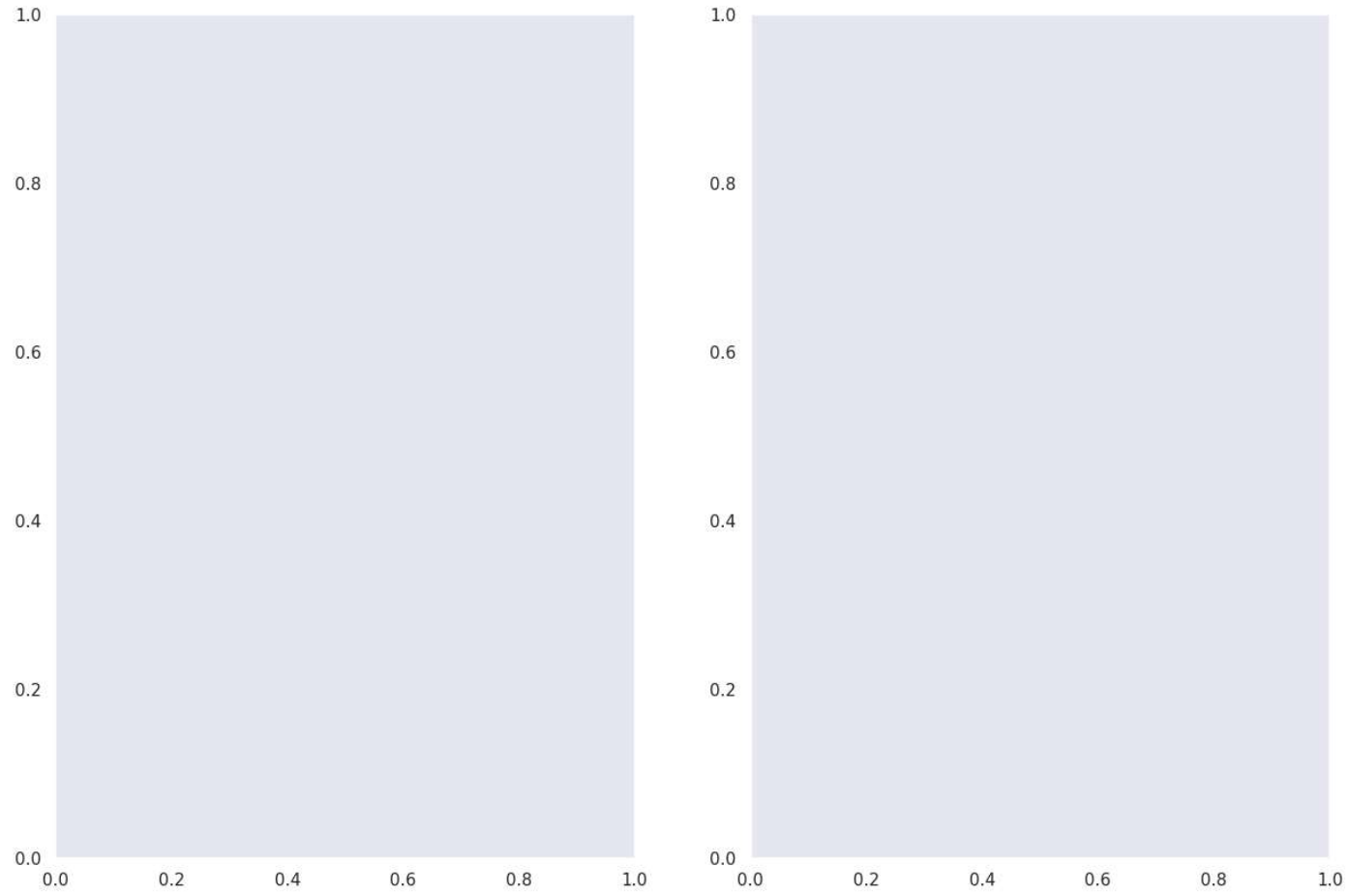
Simple sine plot

```
x=np.linspace(0,10,1000)
plt.plot(x,np.sin(x),x,np.cos(x));
```



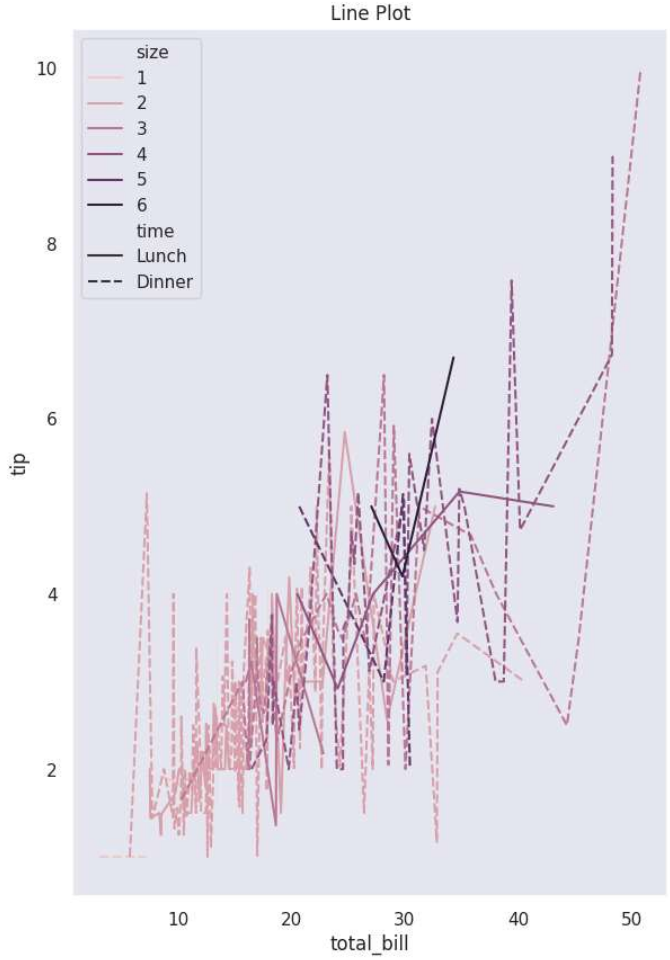

```
#1.Relational plots

sns.set(style="dark")
fig, ax=plt.subplots(ncols=2, nrows=1, figsize=(15,10))
df=sns.load_dataset("tips")
print(df.head())
```

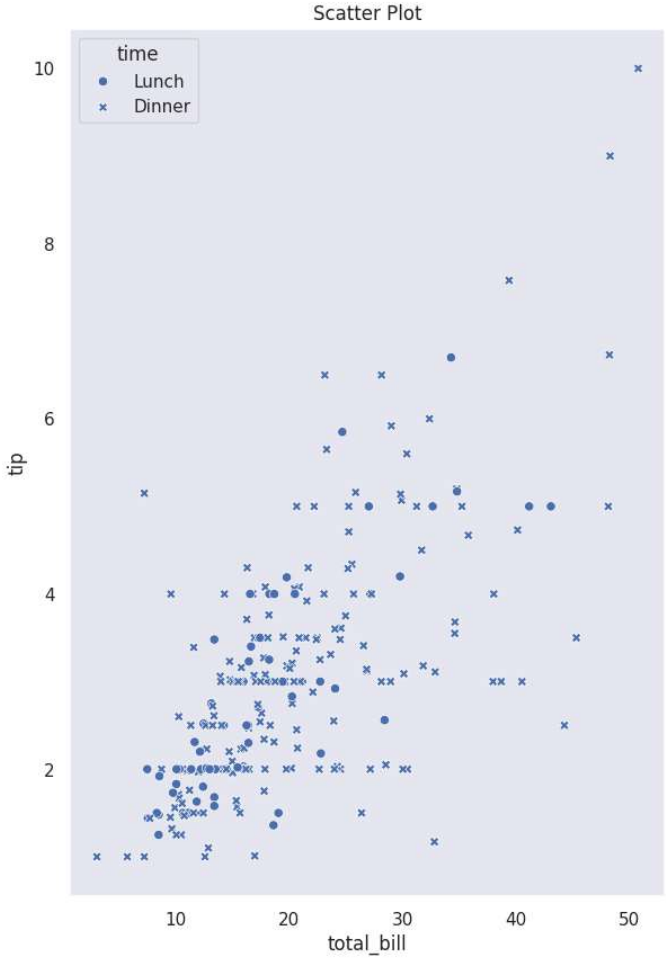
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female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




```
sns.set(style="dark")
fig, ax =plt.subplots(ncols=2, nrows=1, figsize=(15,10))

#Lodaing Data with Seaborn
df=sns.load_dataset("tips")
print(df.head)

#Lineplot
sns.lineplot(x="total_bill", y="tip", hue="size", style="time", data=df,ax=ax[0]).set_title("Line Plot")
#scatterplot
Sct_plt=sns.scatterplot(x="total_bill", y="tip", style="time", data=df,ax=ax[1]).set_title("Scatter Plot")
#Saving Plot
Sct_plt.figure.savefig('Scatter_plot1.png')
print('Plot Saved')
```

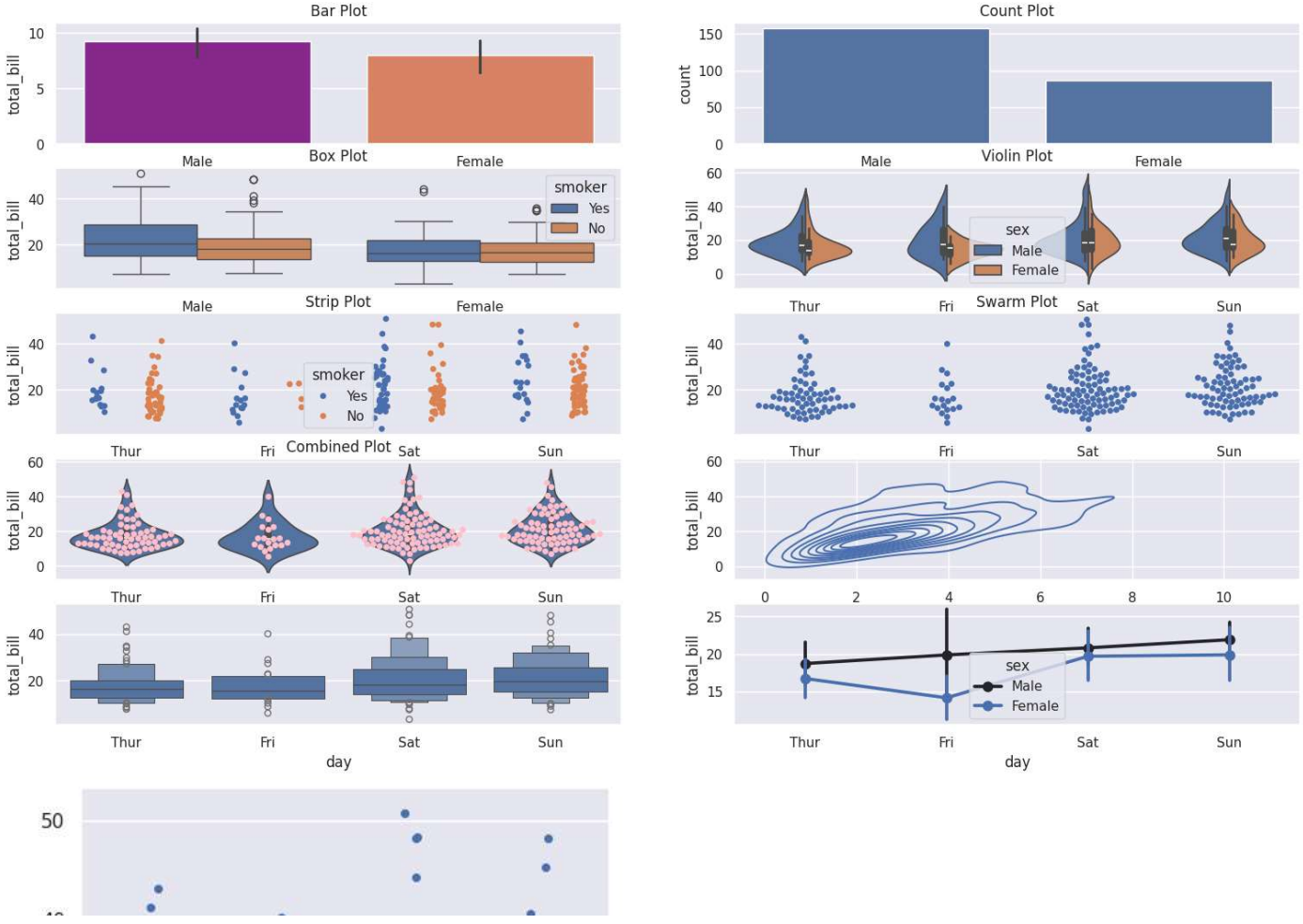
```
<bound method NDFrame.head of      total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   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]>
Plot Saved
```





```
#II. Categorical Plots(barplot,countplot,boxplot)
sns.set_style('darkgrid')
fig, ax=plt.subplots(nrows=5,ncols=2)
fig.set_size_inches(18.5, 10.5)
df=sns.load_dataset('tips')
sns.barplot(x='sex',y='total_bill',data=df,palette='plasma',estimator=np.std,ax=ax[0,0]).set_title('Bar Plot')
sns.countplot(x='sex',data=df,ax=ax[0,1]).set_title('Count Plot')
sns.boxplot(x='sex',y='total_bill',data=df, hue='smoker',ax=ax[1,0]).set_title('Box Plot')
sns.violinplot(x='day',y='total_bill',data=df,hue='sex',split=True,ax=ax[1,1]).set_title('Violin Plot')
sns.stripplot(x='day',y='total_bill',data=df, jitter=True,hue='smoker',dodge= True,ax=ax[2,0]).set_title('Strip Plot')
#Swarm plot similar to strip plot except the fact
#swarmplot
sns.swarmplot(x='day',y='total_bill',data=df,ax=ax[2,1]).set_title("Swarm Plot")
#combing the idea of violinplot and #stripplot
sns.violinplot(x='day',y='total_bill',data=df,ax=ax[3,0])
sns.swarmplot(x='day',y='total_bill',data=df,color ='pink',ax=ax[3,0]).set_title("Combined Plot")
#density plot
sns.kdeplot(x='tip',y='total_bill',data=df,ax=ax[3,1])
#boxenplot
sns.boxenplot(x="day",y="total_bill",color="b",scale="linear",data=df,ax=ax[4,0])
#ridgeplot
sns.pointplot(x="day", y="total_bill",color="b", hue="sex", data=df,ax=ax[4,1])
#catplot
sns.catplot(x='day',y="total_bill",color="b",data=df)
```

```
<ipython-input-8-9faa0f561607>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x='sex',y='total_bill',data=df,palette='plasma',estimator=np.std,ax=ax[0,0]).set_title('Bar Plot')
<ipython-input-8-9faa0f561607>:20: FutureWarning:

The `scale` parameter has been renamed to `width_method` and will be removed in v0.15. Pass `width_method='linear' for the same effect.
  sns.boxenplot(x="day",y="total_bill",color="b",scale="linear",data=df,ax=ax[4,0])
<ipython-input-8-9faa0f561607>:22: FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in v0.14.0. Set `palette='dark:b'` for the same effect.

  sns.pointplot(x="day", y="total_bill",color="b", hue="sex", data=df,ax=ax[4,1])
<seaborn.axisgrid.FacetGrid at 0x7e6072c2f580>
```



iii.**DISTRIBUTION PLOTS** in seaborn is used for examining univariate and bivariate distributions. 4 main types of Distribution plots:

-joinplot

-distploy

-pair plot

-rugplot

```
sns.set_style('whitegrid')
df=sns.load_dataset('iris')
print(df.head())
sns.set_style('whitegrid')
df=sns.load_dataset('iris')
sns.distplot(df['petal_length'],kde=True,color='red',bins=30).set_title('Dist plot')
#jointplot
jointgrid = sns.JointGrid(x='petal_length',y='petal_width',data=df)
jointgrid.plot_joint(sns.scatterplot)
jointgrid.plot_marginals(sns.distplot)
g=sns.jointplot(x='petal_length',y='petal_width',data=df,kind='hex')
g.fig.suptitle('Joint plot')
#pairplot
g=sns.pairplot(df,hue="species",palette='coolwarm')
g.fig.suptitle('pair plot 1')
g.add_legend()
#pairGrid
pairgrid=sns.PairGrid(data=df)
pairgrid=pairgrid.map_offdiag(sns.scatterplot)
pairgrid=pairgrid.map_diag(plt.hist)
#Different kind of plots on upper Triangular Axes,Diagnol Axes and Lower
pairgrid = sns.PairGrid(data=df)
pairgrid = pairgrid.map_upper(sns.scatterplot)
pairgrid = pairgrid.map_diag(plt.hist)
pairgrid=pairgrid.map_lower(sns.kdeplot)
g=sns.PairGrid(df,diag_sharey=False,corner=True)
g.map_lower
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
<ipython-input-9-abcd5699d300>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
#TV matrix plots
fig, ax=plt.subplots(nrows=2, ncols=2, figsize=(15,10))
#data
df1 = sns.load_dataset('flights')
df2=sns.load_dataset('iris')
df11 = pd.pivot_table(values = 'passengers', index ='month', columns ='year', data = df1)
#calculate corlations btw cols in dataframe
dfc1 = df1.corr(numeric_only=True)
dfc2 = df2.corr(numeric_only=True)

sns.heatmap(df11,cmap='YlGnBu', linecolor = 'r', linewidths = 0.5,annot=True,fmt='d',square=True,ax=ax[0,0]).set_title('Heat Map Flights')
sns.heatmap(dfc2,cmap='coolwarm', linecolor = 'black', linewidths = 1, annot=True,ax=ax[0,1]).set_title('Heat Map Iris')

#lower Triangle Display
mask1=np.triu(dfc2)
sns.heatmap(dfc2,annot=True,mask=mask1,ax=ax[1,0],cmap='coolwarm').set_title('Heat map Lower Triangle')
#Upper Triangle Display
mask2=np.tril(dfc2)
sns.heatmap(dfc2,annot=True,cmap='Y1GnBu',mask=mask2,ax=ax[1,1]).set_title('Heat Upper Triangle')
#cluster maps
mask2=np.tril(dfc2)
sns.clustermap(df11,cmap='RdY1Gn')
sns.clustermap(df11,cmap='plasma',standard_scale=1)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-10-4647f9e62818> in <cell line: 19>()
     17 #Upper Triangle Display
```