**Seaborn**

Seaborn is a python data visualization library based on matplotlib. It provides a high level interface for drawing atteractive and informative statistical graphics.

**Seaborn Plots types:**

1. Categorical : Countplot,Barplot,Boxplot,Violinplot,Stripplot,Swarmplot,Factorplot

2. Distribution : Histplot,KDEPlot,RugPlot,ECDPlot,Displot,Joinplot,Pairplot.

3. Regression : simple linear plot, Linear plot with additional parameters, Setting the size and color of the plot, Displaying multiple plots, size and aspect ratio of the plots.

4. Matrix : Heat maps, Cluster maps.

5. Multi-plot Grids.

6. Relational : ScatterPlot,Line Plot,Rel Plot

Standard Import for seaborn :

import seaborn as sns

1. **Countplot:**

syntax:

-----------

sns.countplot(data=None, \*, x=None, y=None, hue=None, order=None, hue\_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, hue\_norm=None, stat='count', width=0.8, dodge='auto', gap=0, log\_scale=None, native\_scale=False, formatter=None, legend='auto', ax=None, \*\*kwargs)

eg:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

s= pd.read\_csv('supermarket\_sales.csv',)

# print(s)

w=sns.countplot(data=s,x='Quantity',)

plt.show()

**Use Cases:**

1. **Create a basic countplot to get number of transaction for each of the product line and change the figure size:**

s= pd.read\_csv('supermarket\_sales.csv',)

print(s.columns)

plt.figure(figsize=(15,5))

w=sns.countplot(data=s,x='Product line',)

plt.show()

2. **Make it horizontal bar plot:**

s= pd.read\_csv('supermarket\_sales.csv',)

print(s.columns)

plt.figure(figsize=(15,5))

w=sns.countplot(data=s,y='Product line',)

plt.show()

3. **Add hue to get the count on two categories i.e product line and Gender.**

s= pd.read\_csv('supermarket\_sales.csv',)

print(s.columns)

plt.figure(figsize=(15,5))

w=sns.countplot(data=s,x='Product line',hue='Gender')

plt.show()

4. **Use different color palette.**

s= pd.read\_csv('supermarket\_sales.csv',)

print(s.columns)

plt.figure(figsize=(15,5))

w=sns.countplot(data=s,x='Product line',hue='Gender',palette='bone')

plt.show()

5. **Change style using facecolor,linewidth and edge color.**

s= pd.read\_csv('supermarket\_sales.csv',)

print(s.columns)

plt.figure(figsize=(15,5))

w=sns.countplot(data=s,x='Product line',facecolor=(1,0,1,0),linewidth=5,edgecolor=sns.color\_palette('dark',3))

plt.show()

2. **BarPlot:**

sns.barplot(data=None, \*, x=None, y=None, hue=None, order=None, hue\_order=None, estimator='mean', errorbar=('ci', 95), n\_boot=1000, seed=None, units=None, weights=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, hue\_norm=None, width=0.8, dodge='auto', gap=0, log\_scale=None, native\_scale=False, formatter=None, legend='auto', capsize=0, err\_kws=None, ci=<deprecated>, errcolor=<deprecated>, errwidth=<deprecated>, ax=None, \*\*kwargs)

1. **Create a basic bar plot:**

s= pd.read\_csv('supermarket\_sales.csv',)

print(s.columns)

plt.figure(figsize=(15,5))

w=sns.barplot(x='Product line',y='Total',data=s)

plt.show()

2. **Add hue to the barplot:**

s= pd.read\_csv('supermarket\_sales.csv',)

print(s.columns)

plt.figure(figsize=(15,5))

w=sns.barplot(x='Product line',y='Total',hue='Gender',data=s)

plt.show()

3. **Make the barplot Horizontal:**

s= pd.read\_csv('supermarket\_sales.csv',)

print(s.columns)

plt.figure(figsize=(15,5))

w=sns.barplot(y='Product line',x='Total',hue='Gender',data=s)

plt.show()

4. **Plot the bars in a Order:**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

x=s['Product line'].sort\_values()

print(x.unique())

w=sns.barplot(x='Product line',y='Total',data=s,order=['Electronic\_accessories','Fashion\_accessories','Food\_and\_beverages',

'Health\_and\_beauty','Home\_and\_lifestyle','Sports\_and\_travel'])

plt.show()

5. **Add CAP on the Error Bar:**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.barplot(x='Product line',y='Total',data=s,capsize=0.2)

plt.show()

6. **Remove the error bar using ci:**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.barplot(x='Product line',y='Total',data=s,capsize=0.2,ci=None)

plt.show()

7. Change Bar Colors by using COLOR attribute:

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.barplot(x='Product line',y='Total',data=s,hue='Gender',color='red')

plt.show()

8. **Change Color using PALLETE attribute:**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.barplot(x='Product line',y='Total',data=s,hue='Gender',color='red',palette='cool')

plt.show()

9. **Use SATURATION parameter:**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.barplot(x='Product line',y='Total',data=s,hue='Gender',color='red',palette='cool',saturation=5)

plt.show()

10. **Change Default Aggregation method using ESTIMATOR Parameter:**

s= pd.read\_csv('supermarket\_sales.csv',)

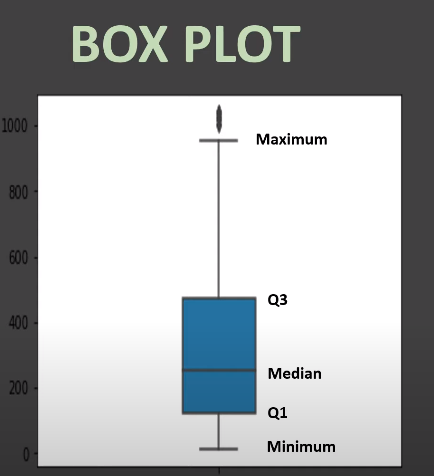
plt.figure(figsize=(15,5))

w=sns.barplot(x='Product line',y='Total',data=s,hue='Gender',estimator=np.median)

plt.show()

3. **Box Plot:**

sns.boxplot(data=None, \*, x=None, y=None, hue=None, order=None, hue\_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, dodge='auto', width=0.8, gap=0, whis=1.5, linecolor='auto', linewidth=None, fliersize=None, hue\_norm=None, native\_scale=False, log\_scale=None, formatter=None, legend='auto', ax=None, \*\*kwargs)



Use Case.

1. **Create a basic boxplot**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.boxplot(y='Total',data=s,width=0.2)

plt.show()

2. **Create a box plot on a one numeric variable by a CATEGORICAL variable.**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.boxplot(y='Total',x='City',data=s,width=0.2)

plt.show()

3. **Create a basic box plot on one numeric variable by TWO CATEGORICAL variable using HUE attribute.**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.boxplot(y='Total',x='City',data=s,hue='Gender')

plt.show()

4. **Add Mean maker in the box plot using showmeans attribute and change its style using meanprops.**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.boxplot(y='Total',x='City',data=s,hue='Gender',showmeans=True,meanprops={"marker":'o', "markerfacecolor":'white',"markersize":"10","markeredgecolor":'black'})

plt.show()

5. **Make HORIZONTAL box Plot.**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.boxplot(x='Total',y='City',data=s,hue='Gender',showmeans=True,meanprops={"marker":'o', "markerfacecolor":'white',"markersize":"10","markeredgecolor":'black'})

plt.show()

6. **Change PALLETE,LINE WIDTH etc.**

s= pd.read\_csv('supermarket\_sales.csv',)

plt.figure(figsize=(15,5))

w=sns.boxplot(y='Total',x='City',data=s,hue='Gender',showmeans=True,linewidth=2,palette='Set3')

plt.show()

7. **Create box plot for EACH of the Numeric variable in the dataframe.**

s= pd.read\_csv('supermarket\_sales.csv',)

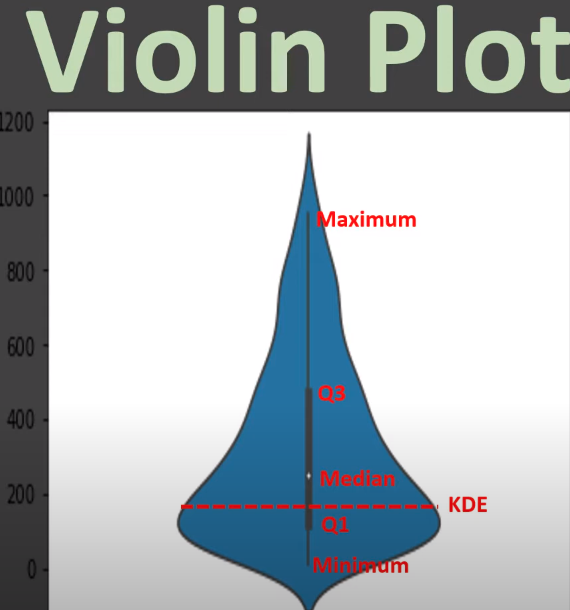
plt.figure(figsize=(15,5))

w=sns.boxplot(data=s)

plt.show()

4. **Violin Plot:**

sns.violinplot(data=None, \*, x=None, y=None, hue=None, order=None, hue\_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, inner='box', split=False, width=0.8, dodge='auto', gap=0, linewidth=None, linecolor='auto', cut=2, gridsize=100, bw\_method='scott', bw\_adjust=1, density\_norm='area', common\_norm=False, hue\_norm=None, formatter=None, log\_scale=None, native\_scale=False, legend='auto', scale=<deprecated>, scale\_hue=<deprecated>, bw=<deprecated>, inner\_kws=None, ax=None, \*\*kwargs)



Use Case:

1. **Create a basic Violin Plot:**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.violinplot(y='Total',data=s)

plt.show()

2. **Create a Violin plot on two categorical and one numeric variable and use split.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.violinplot(y='Total',x='Payment',data=s,hue='Gender',split=True)

plt.show()

3. **Change the box in the violinplot to horizontal lines.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.violinplot(y='Total',x='Payment',data=s,hue='Gender',split=True,inner='stick')

plt.show()

inner : {‘box’,’quart’,’stick’,’point’,None}

4. **Change the amout of smoothing using bw attribute.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.violinplot(y='Total',x='Payment',data=s,hue='Gender',split=True,inner='quartile',bw=2)

plt.show()

5. **Cut out the extreme values.**

s= pd.read\_csv('supermarket\_sales.csv',)

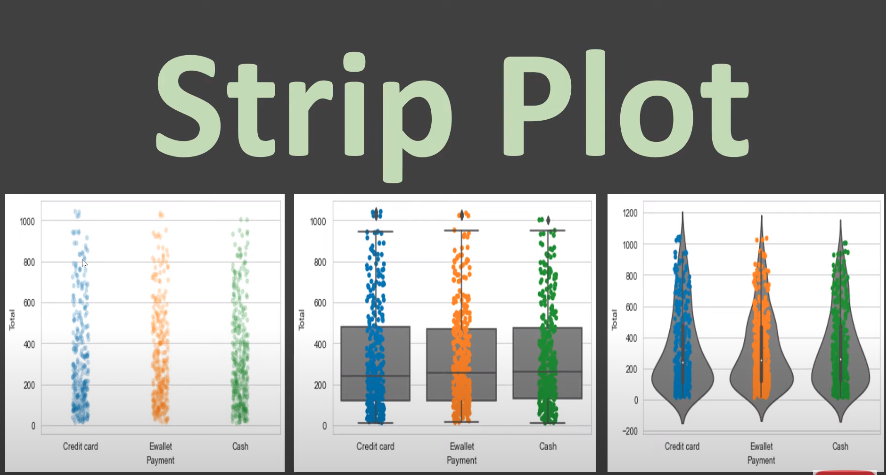
w=sns.violinplot(y='Total',x='Payment',data=s,hue='Gender',split=True,inner='quartile',bw=0.2,cut=0)

plt.show()

5. Strip Plot :

A strip plot is a graphical data anlysis technique for summarizing a univariate data set.

sns.stripplot(data=None, \*, x=None, y=None, hue=None, order=None, hue\_order=None, jitter=True, dodge=False, orient=None, color=None, palette=None, size=5, edgecolor=<default>, linewidth=0, hue\_norm=None, log\_scale=None, native\_scale=False, formatter=None, legend='auto', ax=None, \*\*kwargs)



1. **Create a basic STRIP Plot.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.stripplot(data=s,y='Total')

plt.show()

2. **Expand markers in strip plot using jitter.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.stripplot(data=s,y='Total',jitter=0.2)

plt.show()

3. **Draw line around the points using LINEWIDTH.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.stripplot(data=s,y='Total',linewidth=0.5)

plt.show()

4. **Include third categorial variable with HUE.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.stripplot(data=s,y='Total',x='Payment',linewidth=0.5,hue='Gender',jitter=0.2)

plt.show()

5. **Seprate each level of hue using DODGE.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.stripplot(data=s,y='Total',x='Payment',linewidth=0.5,hue='Gender',jitter=0.2,dodge=True)

plt.show()

6. **Draw the strips on the top of voilin plot.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.stripplot(data=s,y='Total',x='Payment',)

r=sns.violinplot(data=s,y='Total',x='Payment',jitter=0.2,color='grey')

plt.show()

7. **Draw the strips on the top of box plot.**

s= pd.read\_csv('supermarket\_sales.csv',)

w=sns.stripplot(data=s,y='Total',x='Payment',jitter=0.2,color='grey')

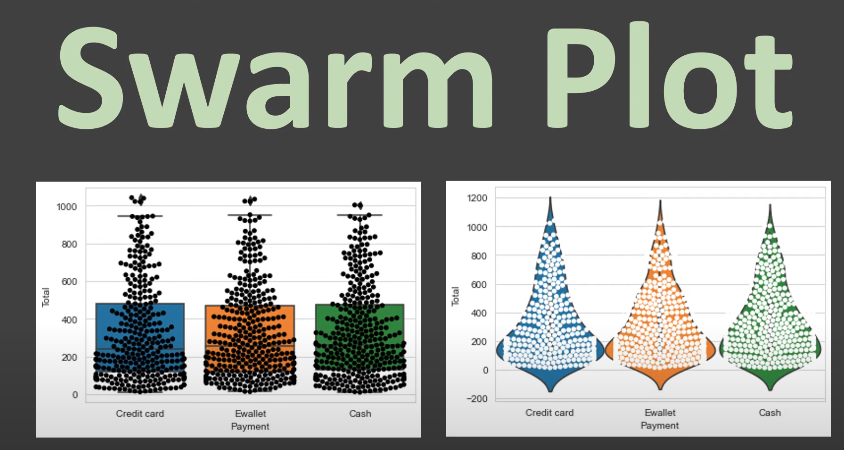
r=sns.boxplot(data=s,y='Total',x='Payment',)

plt.show()

6. **SWARM Plot:**

To represent each of the data points on the plot. In case of strip plot same data points overlaps in the strip plot so that case we have to go for swarm plot.

sns.swarmplot(data=None, \*, x=None, y=None, hue=None, order=None, hue\_order=None, dodge=False, orient=None, color=None, palette=None, size=5, edgecolor=None, linewidth=0, hue\_norm=None, log\_scale=None, native\_scale=False, formatter=None, legend='auto', warn\_thresh=0.05, ax=None, \*\*kwargs)



Use Cases:

1. **Create a basic swarm plot on total column.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.swarmplot(data=s,y='Total')

plt.show()

2. **Create swarm plot group by categories.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.swarmplot(data=s,y='Total',x='Payment')

plt.show()

3. **Showing Hues seperately on the categorical axis.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.swarmplot(data=s,y='Total',x='Payment',hue='Gender',split=True

)

plt.show()

4. **Styling a swarm- change the marker,color,edge color.**

s=pd.read\_csv('supermarket\_sales.csv')

plt.figure(figsize=(15,5))

w=sns.swarmplot(data=s,y='Total',x='Payment',marker='\*',size=10,color='green',edgecolor='black')

plt.show()

5. **Overlay a swarm plot on a box plot.**

s=pd.read\_csv('supermarket\_sales.csv')

t=sns.boxplot(data=s,y='Total',x='Payment')

w=sns.swarmplot(data=s,y='Total',x='Payment',color='black')

plt.show()

6.**Overlay a swarm pot on a violin plot.**

s=pd.read\_csv('supermarket\_sales.csv')

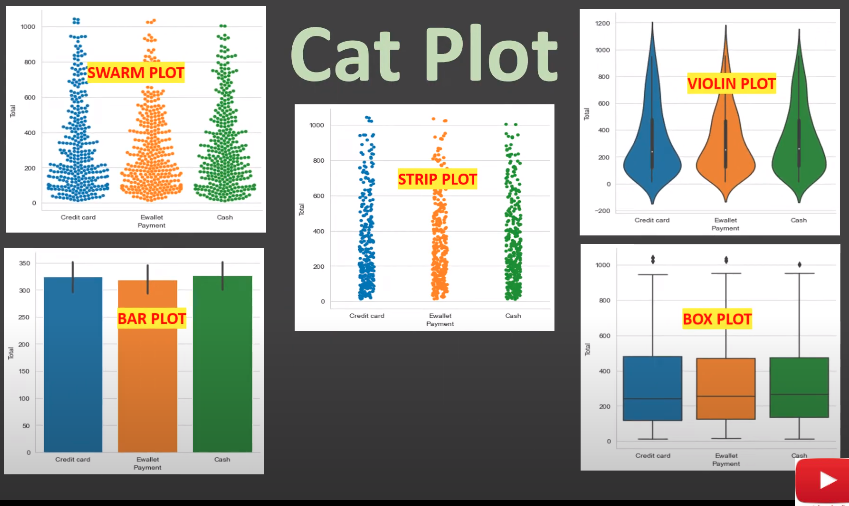
t=sns.violinplot(data=s,y='Total',x='Payment',inner=None,width=1)

w=sns.swarmplot(data=s,y='Total',x='Payment',color='white')

plt.show()

7. Factorplot/ CAT Plot:

In Seaborn, catplot() is a function used to create categorical plots, which are plots that display one or more categorical variables. This function provides a powerful interface for creating various types of categorical plots by specifying the kind of plot through the kind parameter.



syntax:

sns.catplot(data=None, \*, x=None, y=None, hue=None, row=None, col=None, kind='strip', estimator='mean', errorbar=('ci', 95), n\_boot=1000, seed=None, units=None, weights=None, order=None, hue\_order=None, row\_order=None, col\_order=None, col\_wrap=None, height=5, aspect=1, log\_scale=None, native\_scale=False, formatter=None, orient=None, color=None, palette=None, hue\_norm=None, legend='auto', legend\_out=True, sharex=True, sharey=True, margin\_titles=False, facet\_kws=None, ci=<deprecated>, \*\*kwargs)

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.catplot(data=s,x='Payment',y='Total',kind='violin',col='Gender',row='Customer type',palette='PuOr')

plt.show()

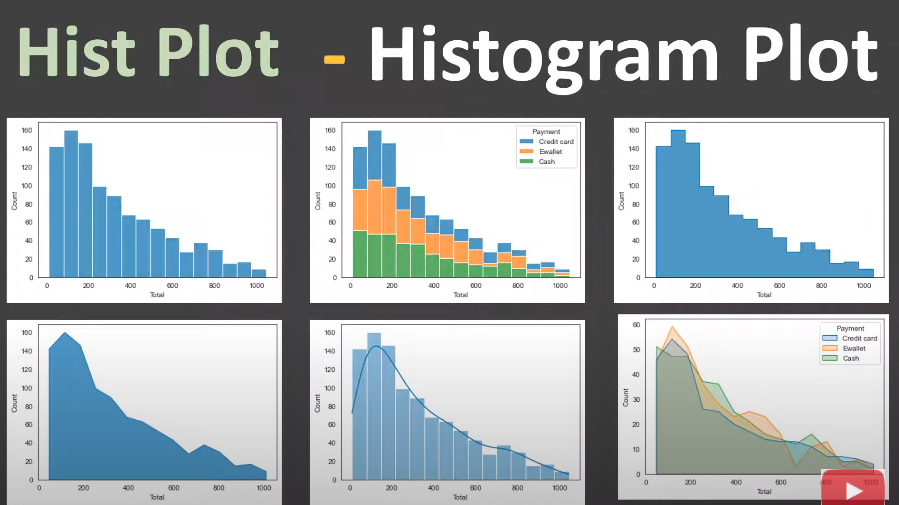
{kind : violin,box,strip}

**Distribution Bars:**

1. Hist Bar:

In Seaborn, histplot() is a function used to create a histogram, which is a graphical representation of the distribution of numerical data. It divides the data into bins and displays the frequency of observations in each bin as bars.

sns.histplot(data=None, \*, x=None, y=None, hue=None, weights=None, stat='count', bins='auto', binwidth=None, binrange=None, discrete=None, cumulative=False, common\_bins=True, common\_norm=True, multiple='layer', element='bars', fill=True, shrink=1, kde=False, kde\_kws=None, line\_kws=None, thresh=0, pthresh=None, pmax=None, cbar=False, cbar\_ax=None, cbar\_kws=None, palette=None, hue\_order=None, hue\_norm=None, color=None, log\_scale=None, legend=True, ax=None, \*\*kwargs)



UseCases:

1. **Create a basic Histogram on x and y axis.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.histplot(data=s,x='Total')

plt.show()

2. **Change the Bins width using binwidth argument.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.histplot(data=s,x='Total',binwidth=150)

plt.show()

3. **Change Number of Bins and Interval.**

s=pd.read\_csv('supermarket\_sales.csv')

plt.figure(figsize=(15,5))

w=sns.histplot(data=s,x='Total',bins=np.arange(0,1100,50))

plt.xticks(np.arange(0,1100,50))

plt.show()

or

w=sns.histplot(data=s,x='Total',bins=5)

4. **Combine with KDE.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.histplot(data=s,x='Total',kde=True)

plt.show()

5. **Use a Categorical in HUE and stack it using multiple argument.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.histplot(data=s,x='Total',hue='Payment',multiple='stack')

plt.show()

6. **Make it a step/poly plot using Element argument and change the FILL.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.histplot(data=s,x='Total',element='step',fill=False)

plt.show()

(or)

w=sns.histplot(data=s,x='Total',hue='Payment',element='poly',)

7. **Use Categorical variable and shrink it.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.histplot(data=s,x='Total',shrink=2,stat='density')

plt.show()

8. **create a Bivariate Histogram.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.histplot(data=s,x='Total',y='gross income')

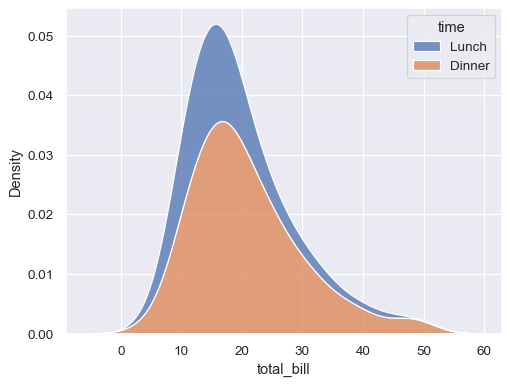
plt.show()

2. KDE Plot

A KDE (Kernel Density Estimation) plot, implemented in Python's Seaborn library, is a visualization technique used to estimate the probability density function of a continuous random variable. It essentially provides a smoothed, continuous representation of the underlying distribution of a dataset.

Syntax:

sns.kdeplot(data=None, \*, x=None, y=None, hue=None, weights=None, palette=None, hue\_order=None, hue\_norm=None, color=None, fill=None, multiple='layer', common\_norm=True, common\_grid=False, cumulative=False, bw\_method='scott', bw\_adjust=1, warn\_singular=True, log\_scale=None, levels=10, thresh=0.05, gridsize=200, cut=3, clip=None, legend=True, cbar=False, cbar\_ax=None, cbar\_kws=None, ax=None, \*\*kwargs)



Use Cases:

1. Create a basic KDE Plot for total column.

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.kdeplot(data=s,x='Total')

plt.show()

2. **Create KDE for all the numeric variables in dataframe.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.kdeplot(data=s,)

plt.show()

3. **Adjust the smoothing using bw\_adjust.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.kdeplot(data=s,x='Total',bw\_adjust=0.2)

plt.show()

4. **Group the KDE on a category variable.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.kdeplot(data=s,x='Total',hue='Payment',)

plt.show()

5. **Stack KDE on a category using multiple argument.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.kdeplot(data=s,x='Total',hue='Payment',multiple='stack')

plt.show()

mulitiple= stack,fill,layer

6. **Use log scaling to map the variable in KDE.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.kdeplot(data=s,x='Total',log\_scale=True)

plt.show()

7. **Change styling of hue KDE using linewidth,palette,alpha etc.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.kdeplot(data=s,x='Total',hue='Payment',multiple='stack',linewidth=5,palette='Dark2',alpha=0.1)

plt.show()

8. **Create a bivariate KDE.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.kdeplot(data=s,x='Unit price',y='gross income')

plt.show()

9. **Group the bivariate KDE on a categorical variable and show the countours.**

s=pd.read\_csv('supermarket\_sales.csv')

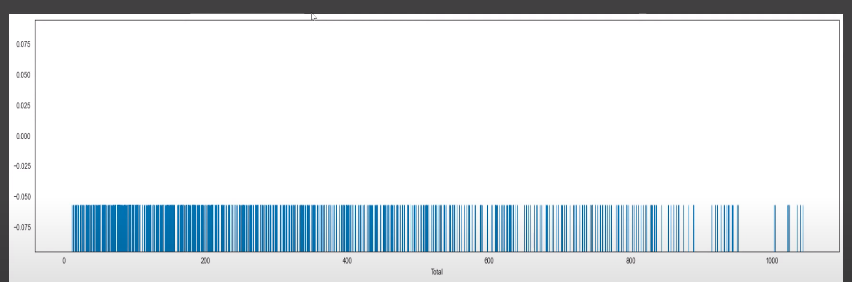
w=sns.kdeplot(data=s,x='Unit price',y='gross income',hue='Gender',fill=True,levels=5,thresh=0.2)

plt.show()

3. **Rug Plot:**

Rugplot is a simple one-dimensional representation of a dataset. It essentially places a small vertical tick mark (or "rug") along the x-axis for each data point in the dataset. These ticks provide a visual indication of the distribution and density of the data points along the axis.

seaborn.rugplot(data=None, \*, x=None, y=None, hue=None, height=0.025, expand\_margins=True, palette=None, hue\_order=None, hue\_norm=None, legend=True, ax=None, \*\*kwargs)



Use-Cases:

1. **Create a basic rug plot for one vairable.**

s=pd.read\_csv('supermarket\_sales.csv')

plt.figure(figsize=(15,5))

w=sns.rugplot(data=s,x='Total')

plt.show()

2. **Create a Rug for two variables.**

s=pd.read\_csv('supermarket\_sales.csv')

plt.figure(figsize=(15,5))

w=sns.rugplot(data=s,x='Total',y='gross income')

plt.show()

3. **Group it by a categorical variable by using hue.**

s=pd.read\_csv('supermarket\_sales.csv')

plt.figure(figsize=(15,5))

w=sns.rugplot(data=s,x='Total',y='gross income',hue='Gender')

plt.show()

4. **Change the height of Rugs in the Rug Plot.**

s=pd.read\_csv('supermarket\_sales.csv')

plt.figure(figsize=(15,5))

w=sns.rugplot(data=s,x='Total',y='gross income',hue='Gender',height=0.05)

plt.show()

5. **Combine this with KDE Plot.**

s=pd.read\_csv('supermarket\_sales.csv')

plt.figure(figsize=(15,5))

r=sns.kdeplot(data=s,x='Unit price',y='gross income',hue='Gender')

w=sns.rugplot(data=s,x='Unit price',y='gross income',hue='Gender')

plt.show()

6. **Combine with a scatter plot.**

s=pd.read\_csv('supermarket\_sales.csv')

plt.figure(figsize=(15,5))

r=sns.scatterplot(data=s,x='Unit price',y='gross income',hue='Gender')

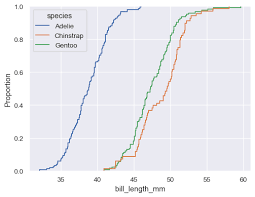
w=sns.rugplot(data=s,x='Unit price',y='gross income',hue='Gender')

plt.show()

4. **ECDF Plot:(Empirical Cumulative Distribution Function)**

The ECDF represents the cumulative distribution of empirical observations in a dataset. It shows the proportion of data points that are less than or equal to each observed value.

sns.ecdfplot(data=None, \*, x=None, y=None, hue=None, weights=None, stat='proportion', complementary=False, palette=None, hue\_order=None, hue\_norm=None, log\_scale=None, legend=True, ax=None, \*\*kwargs)



eg:

----

s=pd.read\_csv('supermarket\_sales.csv')

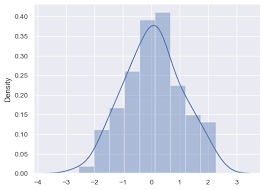
w=sns.ecdfplot(data=s,x='gross income',hue='Gender',stat='count')

plt.show()

5. **Displot:**

In Seaborn, displot() is a versatile function used to create various types of univariate distribution plots. It's a high-level interface that combines functionalities of different seaborn functions such as histplot(), kdeplot(), and ecdfplot().

sns.displot(data=None, \*, x=None, y=None, hue=None, row=None, col=None, weights=None, kind='hist', rug=False, rug\_kws=None, log\_scale=None, legend=True, palette=None, hue\_order=None, hue\_norm=None, color=None, col\_wrap=None, row\_order=None, col\_order=None, height=5, aspect=1, facet\_kws=None, \*\*kwargs)



eg :

s=pd.read\_csv('supermarket\_sales.csv')

sns.displot(data=s,x='Total',hue='Payment',col='Gender',row='Branch')

plt.show()

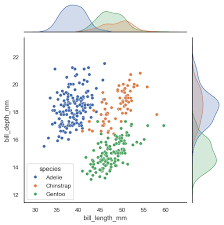
6. **Joint Plot:**

In Seaborn, jointplot() is a function used to create a multi-panel figure that shows the joint distribution between two variables, along with the marginal distributions of each variable. It's particularly useful for exploring the relationship between two continuous variables.

Syntax:

---------

sns.jointplot(data=None, \*, x=None, y=None, hue=None, kind='scatter', height=6, ratio=5, space=0.2, dropna=False, xlim=None, ylim=None, color=None, palette=None, hue\_order=None, hue\_norm=None, marginal\_ticks=False, joint\_kws=None, marginal\_kws=None, \*\*kwargs)



Use Cases:

1. **Drawing a basic joint Plot:**

s=pd.read\_csv('supermarket\_sales.csv')

sns.jointplot(data=s,x='Total',y='gross income')

plt.show()

2. **Change its kind to scatter,kde,hist,hex,reg,resid.**

s=pd.read\_csv('supermarket\_sales.csv')

sns.jointplot(data=s,x='Total',y='gross income',kind='resid')

plt.show()

3. **Grouping based on a categorical variable.**

s=pd.read\_csv('supermarket\_sales.csv')

sns.jointplot(data=s,x='Total',y='gross income',hue='Gender')

plt.show()

4. **Formating All and individiual Plots.**

s=pd.read\_csv('supermarket\_sales.csv')

sns.jointplot(data=s,x='Total',y='gross income',hue='Gender',

color='red',palette='BuPu',joint\_kws=dict(marker='+'),

marginal\_kws=dict(color='green'),)

plt.show()

5. **Plot KDE and RUG on top of joint Plot.**

s=pd.read\_csv('supermarket\_sales.csv')

w=sns.jointplot(data=s,x='Total',y='Unit price',joint\_kws=dict(color='red'))

w.plot\_joint(sns.kdeplot)

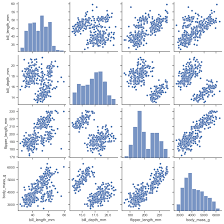
w.plot\_joint(sns.rugplot,height=0.1,color='red')

plt.show()

7. Pair Plot:

In Seaborn, pairplot is a function used to plot pairwise relationships in a dataset. It creates a grid of subplots where each variable in the dataset is plotted against every other variable. It is particularly useful for exploring the correlation between different variables in a dataset and identifying patterns or trends.

sns.pairplot(data, \*, hue=None, hue\_order=None, palette=None, vars=None, x\_vars=None, y\_vars=None, kind='scatter', diag\_kind='auto', markers=None, height=2.5, aspect=1, corner=False, dropna=False, plot\_kws=None, diag\_kws=None, grid\_kws=None, size=None)



USE CASES:

1. **Create a Basic Pair Plot:**

s=pd.read\_csv('supermarket\_sales.csv')

sns.pairplot(data=s)

plt.show()

2. **Changing the non diagonal plot KIND to scatter,kide,hist or reg.**

s=pd.read\_csv('supermarket\_sales.csv')

sns.pairplot(data=s,kind='hist') ---> kind : {‘scatter’,’kde’,’hist’,’reg’}

plt.show()

3. **Adding hue to the pair plot.**

s=pd.read\_csv('supermarket\_sales.csv')

sns.pairplot(data=s,hue='Gender')

plt.show()

4. **Creating PAIR Plot for specific list of vairiables(diag\_kind=None).**

s=pd.read\_csv('supermarket\_sales.csv')

sns.pairplot(data=s,x\_vars=['Total','gross income','cogs'],y\_vars=['Product line','Unit price'],diag\_kind=None)

plt.show()

5. **Showing only the lower triangle using CORNOR argument.**

s=pd.read\_csv('supermarket\_sales.csv')

sns.pairplot(data=s,corner=True)

plt.show()

6. **Making changes specific to Diagonal and Non-Diagonal Plots seperately.**

s=pd.read\_csv('supermarket\_sales.csv')

sns.pairplot(data=s,diag\_kws=dict(color='green',kde=True),plot\_kws=dict(color='red',marker=10,s=100))

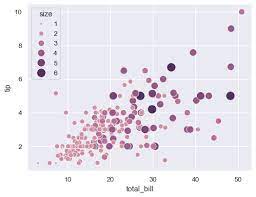
plt.show()

**Relational Plots:**

1. **Scatter Plot:**

In Seaborn, a scatter plot is a type of plot used to visualize the relationship between two numerical variables. It displays individual data points as markers on a two-dimensional plane, with one variable on the x-axis and the other on the y-axis. Scatter plots are useful for identifying patterns, trends, or correlations between variables.

sns.scatterplot(data=None, \*, x=None, y=None, hue=None, size=None, style=None, palette=None, hue\_order=None, hue\_norm=None, sizes=None, size\_order=None, size\_norm=None, markers=True, style\_order=None, legend='auto', ax=None, \*\*kwargs)



Use-Cases:

1. **Create a basic Scatter Plot.**

s=pd.read\_csv('supermarket\_sales.csv').head(50)

w=sns.scatterplot(data=s,x='Unit price',y='cogs')

plt.show()

2. **Grouping basis on a categorical variable using HUE.**

s=pd.read\_csv('supermarket\_sales.csv').head(50)

w=sns.scatterplot(data=s,x='Unit price',y='cogs',hue='Payment')

plt.show()

3. **Grouping basis on a categorical variable using style and styling markers as well.**

s=pd.read\_csv('supermarket\_sales.csv').head(50)

w=sns.scatterplot(data=s,x='Unit price',y='cogs',style='Gender',markers={'Female':'\*','Male':'^'})

plt.show()

4. **Grouping basis on a categorical variable using HUE and Style both together.**

s=pd.read\_csv('supermarket\_sales.csv').head(50)

w=sns.scatterplot(data=s,x='Unit price',y='cogs',style='Gender',markers={'Female':'\*','Male':'^'},hue='Payment')

plt.show()

5. **Grouping basis on numeric variable using HUE and use PALETTE.**

s=pd.read\_csv('supermarket\_sales.csv').head(50)

w=sns.scatterplot(data=s,x='Unit price',y='cogs',style='Gender',markers={'Female':'\*','Male':'^'},hue='Payment',sizes=(20,200),palette='Purples\_r',legend='full')

plt.show()

6. **Change the Marker size with s argument.**

s=pd.read\_csv('supermarket\_sales.csv').head(50)

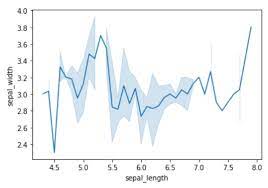
w=sns.scatterplot(data=s,x='Unit price',y='cogs',s=200,)

plt.show()

2. **Line Plot:**

In Seaborn, lineplot is a function used to draw a line plot of one variable against another. It's particularly useful for visualizing the relationship between two variables over a continuous interval.

sns.lineplot(data=None, \*, x=None, y=None, hue=None, size=None, style=None, units=None, weights=None, palette=None, hue\_order=None, hue\_norm=None, sizes=None, size\_order=None, size\_norm=None, dashes=True, markers=None, style\_order=None, estimator='mean', errorbar=('ci', 95), n\_boot=1000, seed=None, orient='x', sort=True, err\_style='band', err\_kws=None, legend='auto', ci='deprecated', ax=None, \*\*kwargs)



Use Case:

1. **Create a basic line Plot and try different estimators.**

s=pd.read\_csv('mart\_linePlot.csv').head(50)

sns.lineplot(data=s,x='Outlet\_Year',y='Sales',ci=None)

plt.show()

sns.lineplot(data=s,x='Outlet\_Year',y='Sales',ci=None,estimator=sum)

2. **Test with different confidence intervals and with different number of Bootstraping.**

s=pd.read\_csv('mart\_linePlot.csv').head(50)

sns.lineplot(data=s,x='Outlet\_Year',y='Sales',ci='sd')

plt.show()

3. **Grouping using HUE and use different Paletters.**

s=pd.read\_csv('mart\_linePlot.csv')

sns.lineplot(data=s,x='Outlet\_Year',y='Sales',ci=None,hue='Outlet\_Size',palette='Accent')

plt.show()

4. **Grouping using Style,use different marker and dashes.**

s=pd.read\_csv('mart\_linePlot.csv')

sns.lineplot(data=s,x='Outlet\_Year',y='Sales',ci=None,style='Outlet\_Size',markers=True,dashes=False)

plt.show()

5. **Grouping using size.**

s=pd.read\_csv('mart\_linePlot.csv')

sns.lineplot(data=s,x='Outlet\_Year',y='Sales',ci=None,size='Outlet\_Size',sizes=(-5,10))

plt.show()

6. **Use Different styling parameters on same or different variables.**

s=pd.read\_csv('mart\_linePlot.csv')

sns.lineplot(data=s,x='Outlet\_Year',y='Sales',ci=None,hue='Outlet\_Size',size='Tier')

plt.show()