Matplotlib:

Basic:

Title:

```
plt.title("Critic vs User Score", fontsize=14, fontweight='bold')
```

Labels:

```
plt.xlabel("Critic Score (0-10)", fontsize=12)

plt.ylabel("User Score (0-10)", fontsize=12)
```

Axis size:

```
plt.xlim(0, 10)

plt.ylim(0, 10)
```

Legend:

```
plt.legend(title="Game platform", loc="best")
```

loc: best-upper right-upper left-lower left-lower right-right-center left-center right-lower center-upper center.

title is hue default

Annotate:

```
plt.annotate("Top Rated", xy=(6.7,7.4), xytext=(4,9),

        arrowprops=dict(facecolor='black', shrink=0))
```

xy: meaning the location on the plot where **x = 6.7** and **y = 7.4**.

xytext: meaning the location on the plot where the annotation text is placed.

Arrowprops-shrink: 0: long , 0.05: middle , 0.2: little.

Grid:

```
plt.grid(True, linestyle='-', alpha=1)
```

Background:

```
plt.gca().set_facecolor("lightblue")
```

Plot Line :

```
x_vals = np.linspace(0, 10)
```

```
plt.plot(x_vals, x_vals, color='red', linestyle='-', linewidth=2)
```

linspace(start,stop).

plt.plot(x,y).

Vlines:

```
plt.vlines(x=df['Critic_Score2'], ymin=0, ymax=df['User_Score'],

        colors='gray', linestyles='-', linewidth=0.1)
```

x: plot line in Critic_score location from 0-10axis

ymin: start location .

ymax: finish location from 0-10 axis.

Hlines:

```
plt.hlines(y=df['User_Score'], xmin=0,
xmax=df['Critic_Score2'],  colors='gray', linestyles='-', linewidth=0.1)
```

Xticks:

```
plt.xticks(rotation=45,fontsize=12)
```

Yticks:

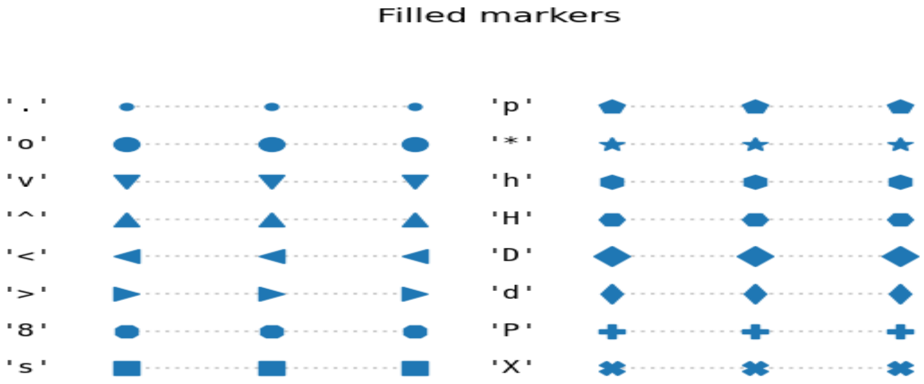
```
plt.yticks(rotation=45,fontsize=12)
```

Figsize:

plt.figure(figsize=(1,100),dpi=100)   : you have put it before plot

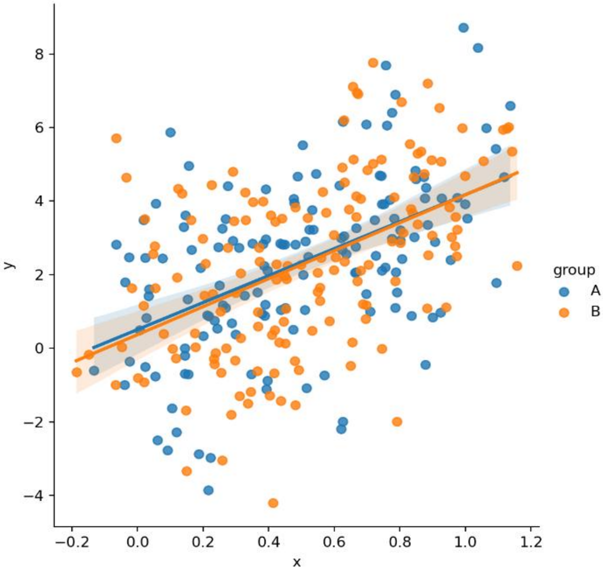
Libraries:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from mpl_toolkits.mplot3d import Axes3D

import mplfinance as mpf
```

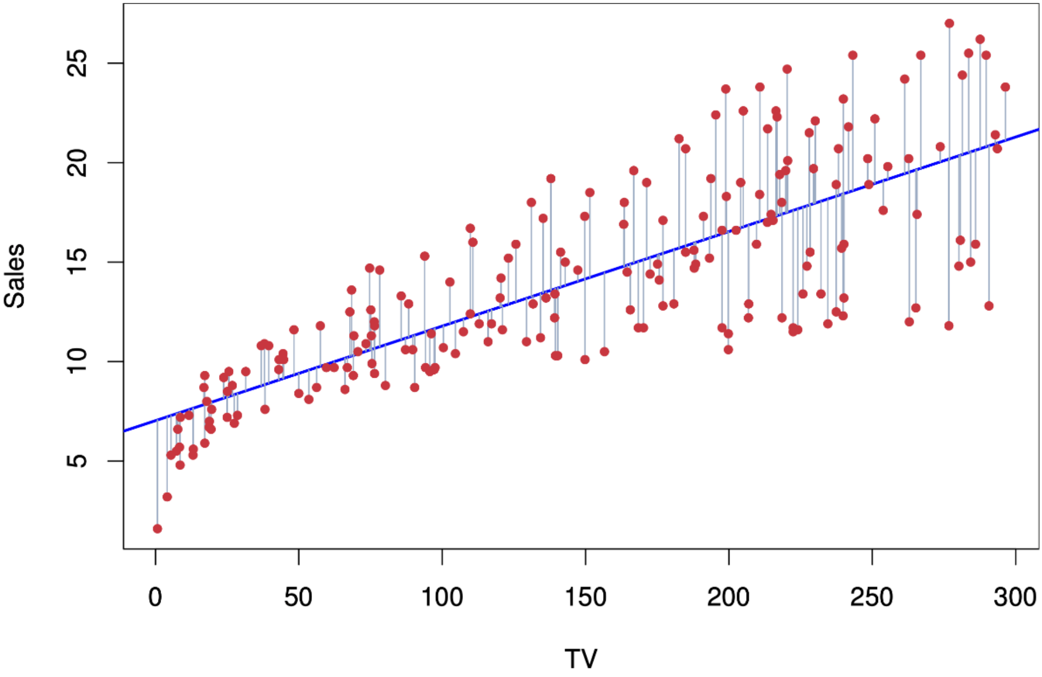


Scatter Plot:

```
sns.scatterplot(
  data=df,
  x='Critic_Score2',
  y='User_Score',
  marker='o',
  hue='Year_of_Release',
  alpha=1,
  legend='full')
```

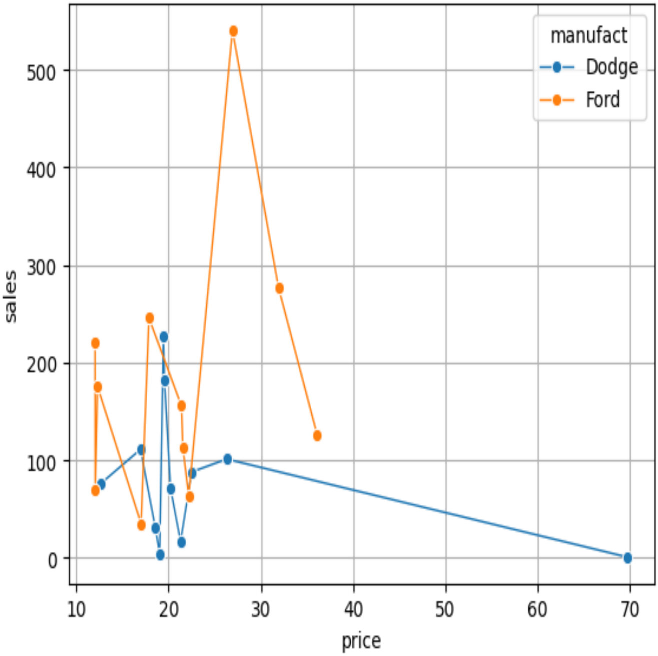


hue: hue means grouping by category.you can edit it with Legend.



Line Plot:

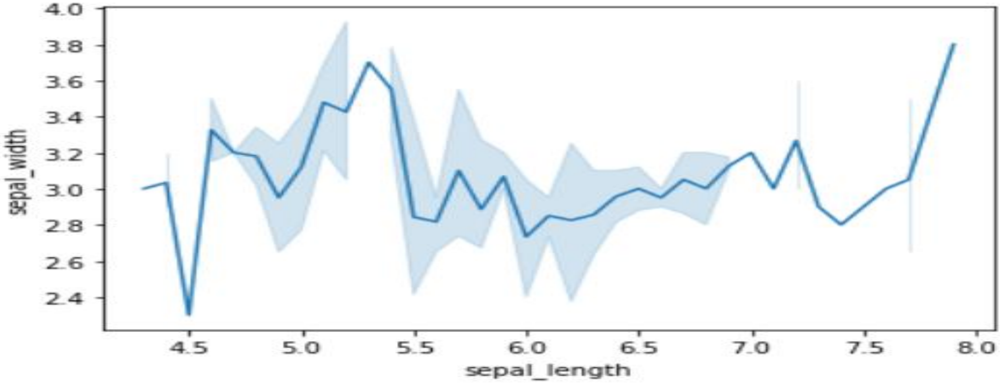
```
sns.lineplot(
  data=selcar,
  x='price',
  y='sales',
  hue='manufact',
  estimator=np.mean,
  ci=95,
  sort=True,
  linewidth=1,
  linestyle='-',
  marker='o'
)
```



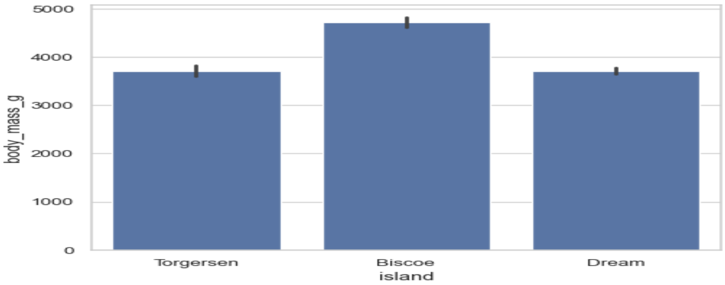
sort=False: Seaborn will plot the data in the order it appears in your DataFrame. If your x-values are not ordered, the line may zig-zag or look messy.

sort=True: Seaborn sorts the x-values before plotting, ensuring the line moves smoothly from left to right. This is especially useful when your data is not already sorted by the x-axis variable.

estimator=np.mean: then our data is some float it convert it to int to plot better.



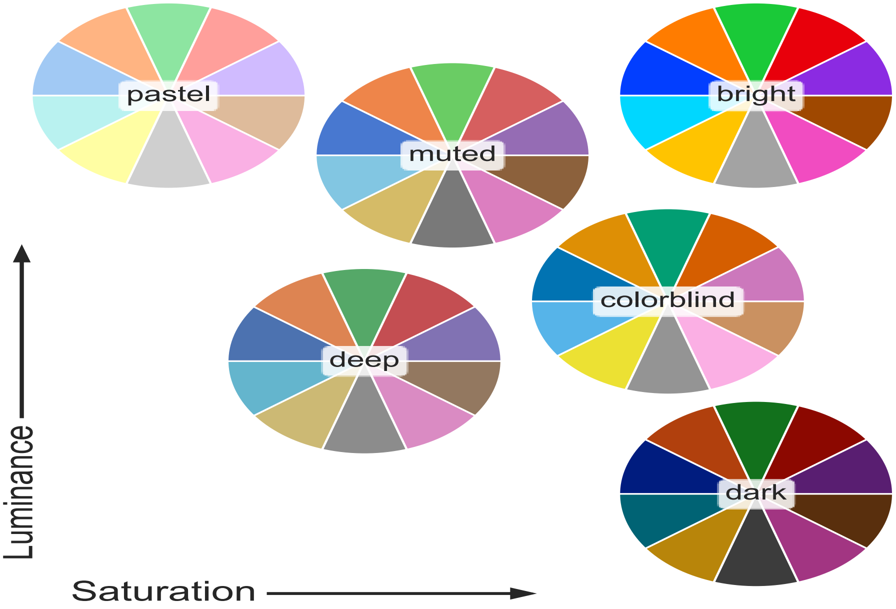
Bar Plot:



```
sns.barplot(
   data=selcar,
   x='model',
   y='sales',
   palette='bright',
   #order=['cat1','cat2','cat3']
   orient='v',
   width=0.8
)
```

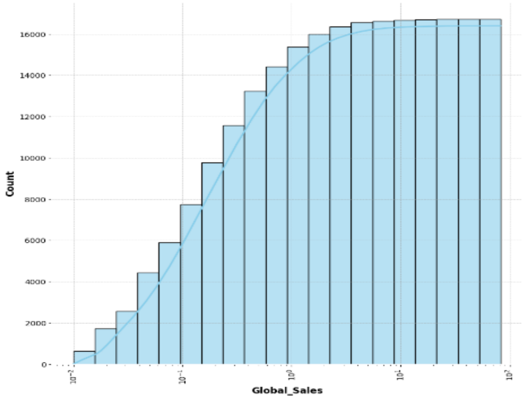
Orient: v = Vertical , h = horizontal.

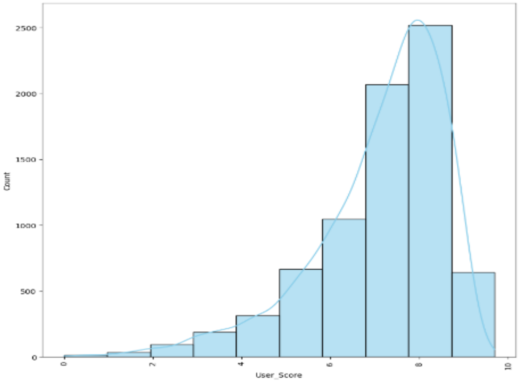
Order: order by cat from first to finally cat.



Histogram:

```
sns.histplot(
  dff['User_Score'],
  bins=100,
  kde=True,
  stat='count',
  color='skyblue',
  alpha=0.6,
  cumulative=False,
  log_scale=False
)
```





Cumulative: False :shows frequency in each bin normally.

Cumulative: True: shows cumulative frequency each bin adds up all previous bins.

Count stat: shows number of data points.

probability / percent stat: shows chance or percentage.

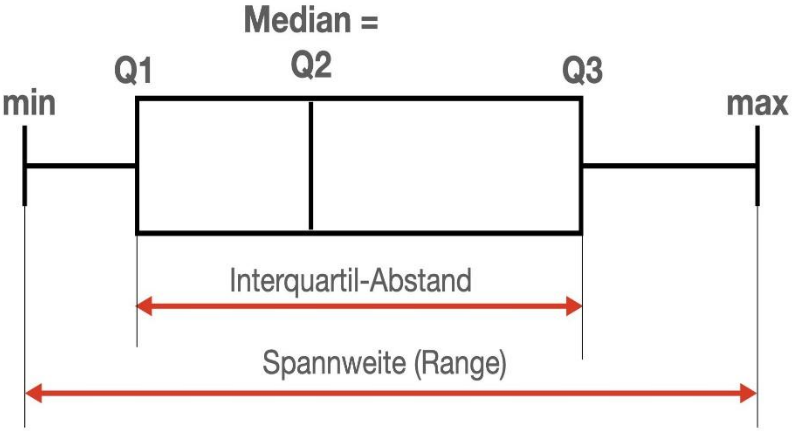
density stat: compares with continuous distributions or KDE.

KDE: Density diagram.

Log_scale: Logarithmic scale.

Box Plot:

```
sns.boxplot(
   data=car,
   x='manufact',
   y='sales',
   showfliers=True,
   notch=False,
   width=0.6,
   orient='v'
)
```



notch: (confidence interval for median).

Showfliers: Hide or show outliers.

Q1 → (Q1 / 25th percentile(25%))

Q3 → (Q3 / 75th percentile(75%))

Median → (Q2 / 50th percentile(50%))

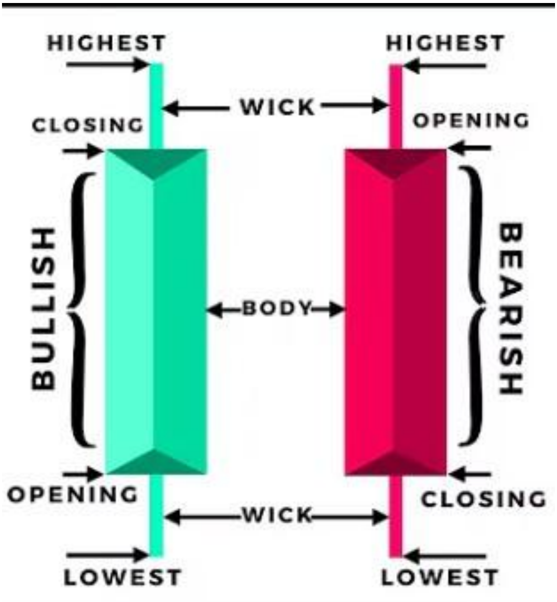
Max → Q3 + 1.5*IQR

Min →Q1 - 1.5*IQR

IQR → Q3 - Q1

You can clean data set outliers with datas that are out of max and min.

Candlestick:



```
mpf.plot(
  dfb,
  type='candle',
  style='charles',
  #volume=True,
  title='Bitcoin Candlestick',
  ylabel='Price ($)',
  ylabel_lower='Volume'
)
```

Index: you have to set inex to date column :

```
dfb['date'] = pd.to_datetime(dfb['date'], dayfirst=True)
dfb.set_index('date', inplace=True)
```

Dataset: must contain Open-Close-Low-High with just these names (if columns named like open_store or … you have to rename them to right name) and,

Volume is Optional to use.(it` s name is important to use in dataset).

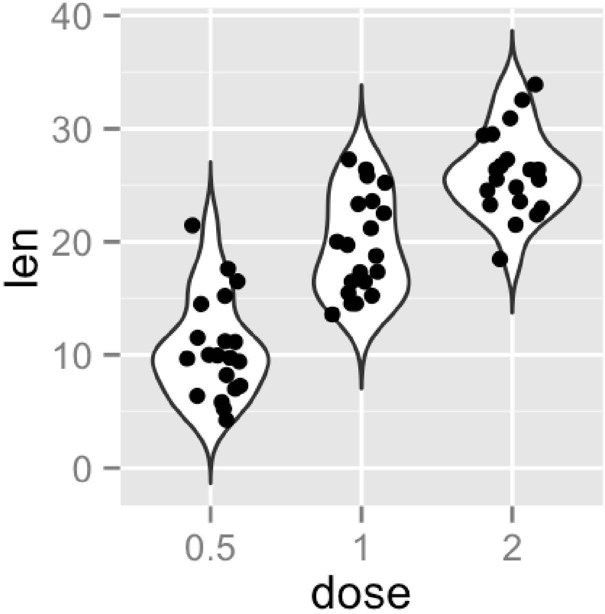
Volume: shows how many units of an asset were traded in a given time period and indicates the strength or validity of a price trend.

Rename:

```
dfb.rename(columns={'vol':'Volume'}, inplace=True)
```

Violin Plot:

```
sns.violinplot(
   data=car,
   x='manufact',
   y='sales',
   split=True,
   inner='quartile',
   scale='width',
   bw=0.2
   orient='v'
)
```



split: divide into two parts.

inner='box': Shows a small boxplot inside the violin.

inner='quartile': Shows only three horizontal lines.

inner='point': Shows only a single dot.

inner='stick': Shows many thin vertical lines.

inner=None: Shows nothing inside.

scale='area': All violins have the same total area.

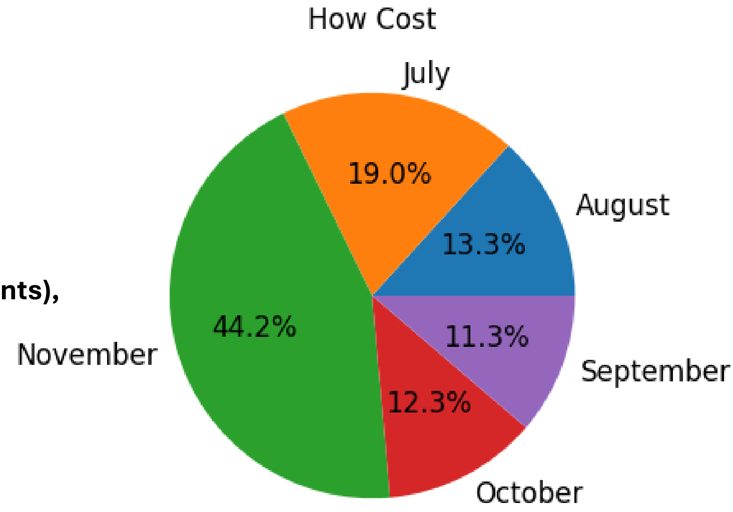
scale='count': Width depends on number of data points.

scale='width': All violins have the same maximum width.

bw: Bandwidth smoothing.

Pie:

```
counts = ex['Category'].value_counts()

plt.pie(

   counts,

   labels=counts.index,

   autopct='%1.1f%%',

   explode=[0.1]*len(counts),

   shadow=True,

   startangle=90

)
```



counts = ex['Category'].value_counts(): need to make them to category.

counts: categories names.

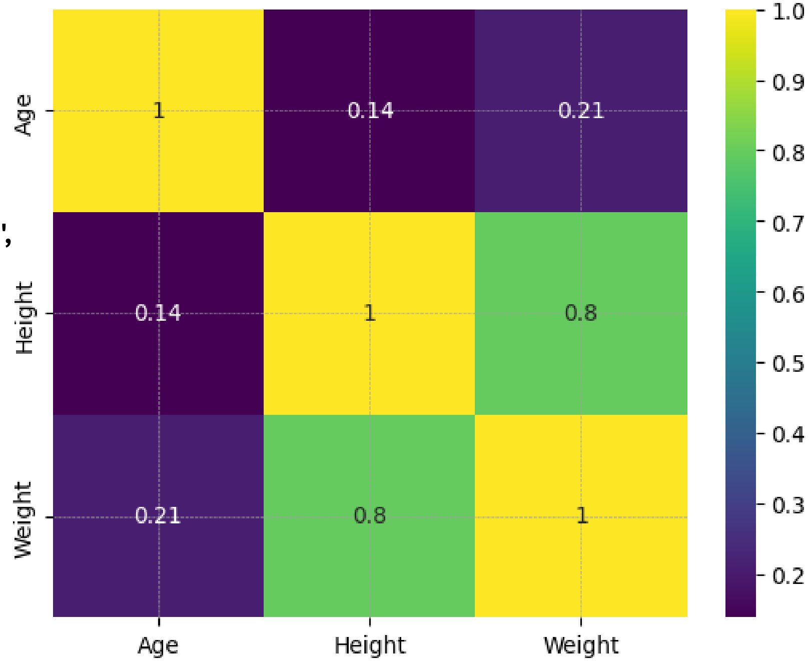
autopct='%1.1f%%': show percent .

explode=[0.1]*len(counts): distance amount.

startangle: angle amount.

Heatmap:

```
dfh = ath[['Age','Height','Weight']].corr()

sns.heatmap(

   dfh,

   annot=True,

   cmap='viridis',

   vmin=-1,

   vmax=1,

   linewidths=1,

   square=True,

   fmt='.2f'

)
```



annot: show percentages.

cmap: palletes color: viridis_coolwarm.

vmin_vmax: color max and min.

fmt: number format

Age and Age: **1.00** : Perfect correlation with itself

Age and Height: **0.14** : Weak positive correlation

Age and Weight: **0.2** : Weak positive correlation

Height and Height: **1.00** : Perfect correlation with itself

Height and Weight: **0.80** : Strong positive correlation

Weight and Weight: **1.00** : Perfect correlation with itself