Python Libraries

16 August 2022 11:11

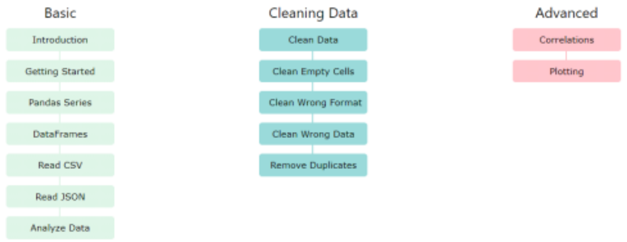
Pandas

- Pandas is a Python library used for working with data sets
- It has functions for analysing, cleaning, exploring, and manipulating data.
- it offers data structures and operations for manipulating numerical tables and time series.

Series - Column of Table
- 1-D Array

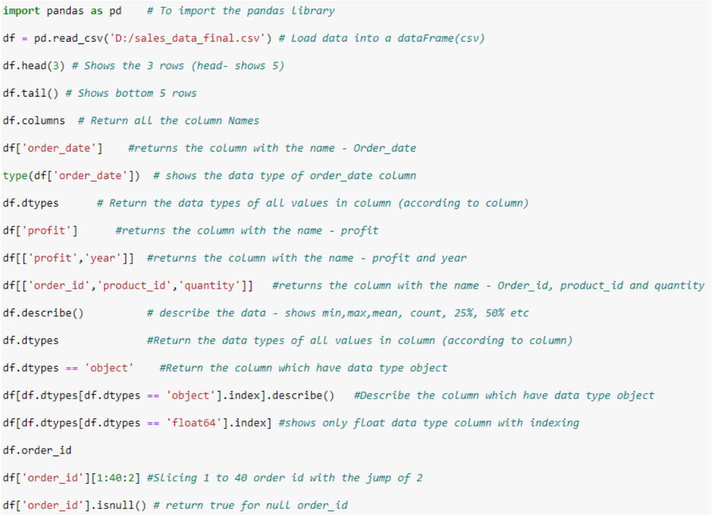
Data Frame- Multi-Dimension Tables

Series is like a column, a Data Frame is the whole table.

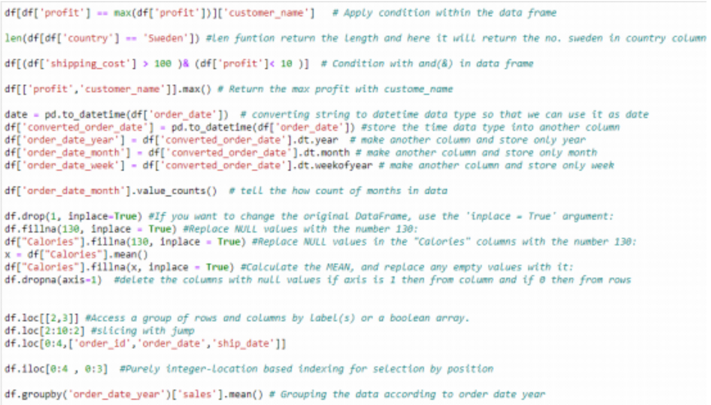

Basic
Introduction
Getting Started
Pandas Series
DataFrames
Read CSV
Read JSON
Analyze Data

Cleaning Data
Clean Data
Clean Empty Cells
Clean Wrong Format
Clean Wrong Data
Remove Duplicates

Advanced
Correlations
Plotting



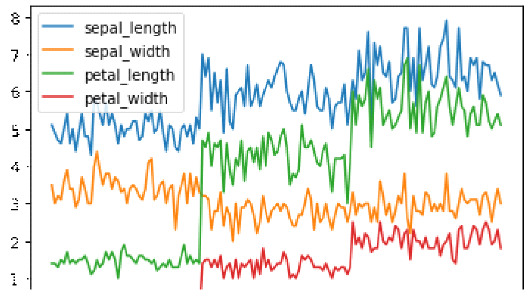
```
import pandas as pd    # To import the pandas library

df = pd.read_csv('D:/sales_data_final.csv') # Load data into a dataFrame(csv)

df.head(3) # Shows the 3 rows (head- shows 5)

df.tail() # Shows bottom 5 rows

df.columns  # Return all the column Names

df['order_date']    #returns the column with the name - Order_date

type(df['order_date'])  # shows the data type of order_date column

df.dtypes      # Return the data types of all values in column (according to column)

df['profit']      #returns the column with the name - profit

df[['profit','year']]  #returns the column with the name - profit and year

df[['order_id','product_id','quantity']]   #returns the column with the name - Order_id, product_id and quantity

df.describe()          # describe the data - shows min,max,mean, count, 25%, 50% etc

df.dtypes              #Return the data types of all values in column (according to column)

df.dtypes == 'object'    #Return the column which have data type object

df[df.dtypes[df.dtypes == 'object'].index].describe()   #Describe the column which have data type object

df[df.dtypes[df.dtypes == 'float64'].index] #shows only float data type column with indexing

df.order_id

df['order_id'][1:40:2] #Slicing 1 to 40 order id with the jump of 2

df['order_id'].isnull() # return true for null order_id
```



```
df[df['profit'] == max(df['profit'])]['customer_name']   # Apply condition within the data frame

len(df[df['country'] == 'Sweden']) #len funtion return the length and here it will return the no. sweden in country column

df[(df['shipping_cost'] > 100 )& (df['profit']< 10 )]  # Condition with and(&) in data frame

df[['profit','customer_name']].max() # Return the max profit with custome_name

date = pd.to_datetime(df['order_date'])  # converting string to datetime data type so that we can use it as date
df['converted_order_date'] = pd.to_datetime(df['order_date']) #store the time data type into another column
df['order_date_year'] = df['converted_order_date'].dt.year  # make another column and store only year
df['order_date_month'] = df['converted_order_date'].dt.month # make another column and store only month
df['order_date_week'] = df['converted_order_date'].dt.weekofyear # make another column and store only week

df['order_date_month'].value_counts()  # tell the how count of months in data

df.drop(1, inplace=True) #If you want to change the original DataFrame, use the 'inplace = True' argument:
df.fillna(130, inplace = True) #Replace NULL values with the number 130:
df["Calories"].fillna(130, inplace = True) #Replace NULL values in the "Calories" columns with the number 130:
x = df["Calories"].mean()
df["Calories"].fillna(x, inplace = True) #Calculate the MEAN, and replace any empty values with it:
df.dropna(axis=1)  #delete the columns with null values if axis is 1 then from column and if 0 then from rows


df.loc[[2,3]] #Access a group of rows and columns by label(s) or a boolean array.
df.loc[2:10:2] #slicing with jump
df.loc[0:4,['order_id','order_date','ship_date']]

df.iloc[0:4 , 0:3]  #Purely integer-location based indexing for selection by position

df.groupby('order_date_year')['sales'].mean() # Grouping the data according to order date year
```

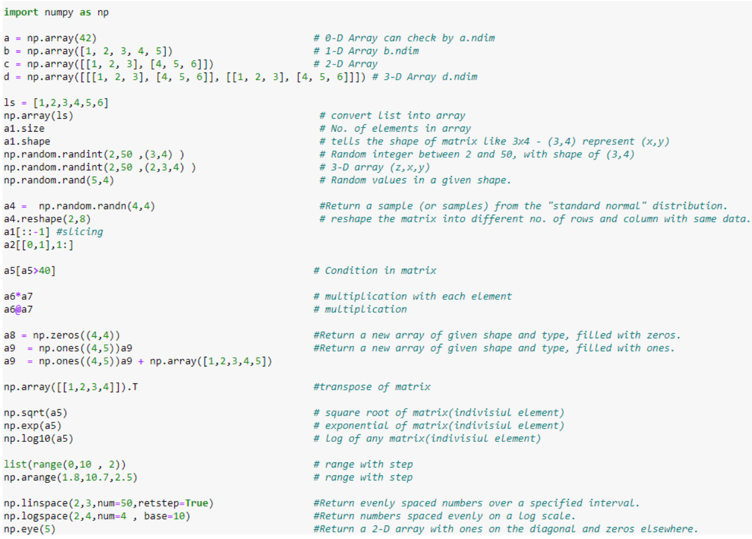
Merge - left will do Later

df.plot()



NumPy

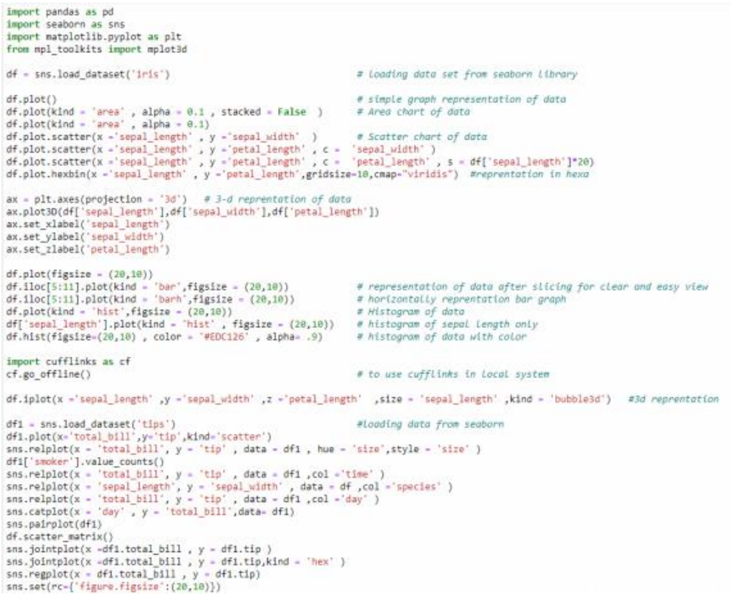
- NumPy is Python Library used for working with arrays.
- NumPy stands for Numerical Python, used in linear algebra and Matrices.
- Array object in NumPy is called ndarray, 50x faster than list.



```
import numpy as np

a = np.array(42)                                        # 0-D Array can check by a.ndim
b = np.array([1, 2, 3, 4, 5])                           # 1-D Array b.ndim
c = np.array([[1, 2, 3], [4, 5, 6]])                    # 2-D Array
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]]) # 3-D Array d.ndim

ls = [1,2,3,4,5,6]
np.array(ls)                                             # convert list into array
a1.size                                                  # No. of elements in array
a1.shape                                                 # tells the shape of matrix like 3x4 - (3,4) represent (x,y)
np.random.randint(2,50 ,(3,4) )                          # Random integer between 2 and 50, with shape of (3,4)
np.random.randint(2,50 ,(2,3,4) )                        # 3-D array (z,x,y)
np.random.rand(5,4)                                      # Random values in a given shape.

a4 =  np.random.randn(4,4)                               #Return a sample (or samples) from the "standard normal" distribution.
a4.reshape(2,8)                                          # reshape the matrix into different no. of rows and column with same data.
a1[::-1] #slicing
a2[[0,1],1:]

a5[a5>40]                                               # Condition in matrix

a6*a7                                                   # multiplication with each element
a6@a7                                                   # multiplication

a8 = np.zeros((4,4))                                    #Return a new array of given shape and type, filled with zeros.
a9  = np.ones((4,5))a9                                  #Return a new array of given shape and type, filled with ones.
a9  = np.ones((4,5))a9 + np.array([1,2,3,4,5])

np.array([[1,2,3,4]]).T                                 #transpose of matrix

np.sqrt(a5)                                             # square root of matrix(indivisiul element)
np.exp(a5)                                              # exponential of matrix(indivisiul element)
np.log10(a5)                                            # log of any matrix(indivisiul element)

list(range(0,10 , 2))                                   # range with step
np.arange(1.8,10.7,2.5)                                 # range with step

np.linspace(2,3,num=50,retstep=True)                    #Return evenly spaced numbers over a specified interval.
np.logspace(2,4,num=4 , base=10)                        #Return numbers spaced evenly on a log scale.
np.eye(5)                                               #Return a 2-D array with ones on the diagonal and zeros elsewhere.
```

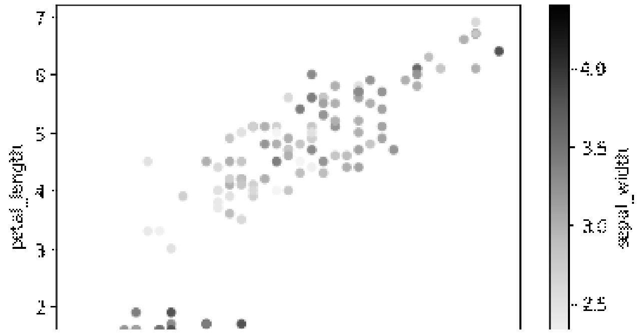
Visualization Libraries

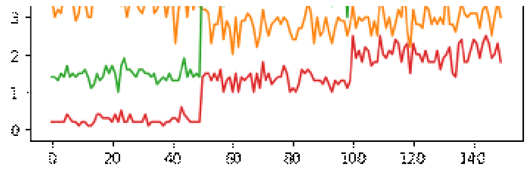
Matplotlib		Seaborn		Cufflinks

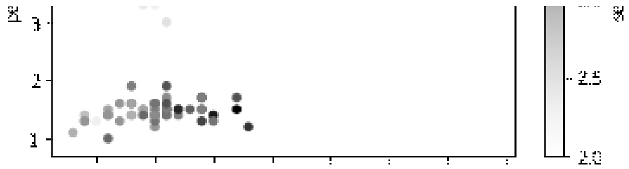


```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

df = sns.load_dataset('iris')                                    # Loading data set from seaborn library

df.plot()                                                        # simple graph representation of data
df.plot(kind = 'area' , alpha = 0.1 , stacked = False  )         # Area chart of data
df.plot(kind = 'area' , alpha = 0.1)
df.plot.scatter(x ='sepal_length' , y ='sepal_width'  )          # Scatter chart of data
df.plot.scatter(x ='sepal_length' , y ='petal_length' , c =  'sepal_width' )
df.plot.scatter(x ='sepal_length' , y ='petal_length' , c =  'petal_length' , s = df['sepal_length']*20)
df.plot.hexbin(x ='sepal_length' , y ='petal_length',gridsize=10,cmap="viridis")  #reprentation in hexa

ax = plt.axes(projection = '3d')   # 3-d reprentation of data
ax.plot3D(df['sepal_length'],df['sepal_width'],df['petal_length'])
ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
ax.set_zlabel('petal_length')

df.plot(figsize = (20,10))
df.iloc[5:11].plot(kind = 'bar',figsize = (20,10))               # representation of data after slicing for clear and easy view
df.iloc[5:11].plot(kind = 'barh',figsize = (20,10))              # horizontally reprentation bar graph
df.plot(kind = 'hist',figsize = (20,10))                         # Histogram of data
df['sepal_length'].plot(kind = 'hist' , figsize = (20,10))       # histogram of sepal length only
df.hist(figsize=(20,10) , color = '#EDC126' , alpha= .9)         # histogram of data with color

import cufflinks as cf
cf.go_offline()                                                  # to use cufflinks in local system

df.iplot(x ='sepal_length' ,y ='sepal_width' ,z ='petal_length'  ,size = 'sepal_length' ,kind = 'bubble3d')   #3d reprentation

df1 = sns.load_dataset('tips')                                   #loading data from seaborn
df1.plot(x='total_bill',y='tip',kind='scatter')
sns.relplot(x = 'total_bill', y = 'tip' , data = df1 , hue = 'size',style = 'size' )
df1['smoker'].value_counts()
sns.relplot(x = 'total_bill', y = 'tip' , data = df1 ,col ='time' )
sns.relplot(x = 'sepal_length', y = 'sepal_width' , data = df ,col ='species' )
sns.relplot(x = 'total_bill', y = 'tip' , data = df1 ,col ='day' )
sns.catplot(x = 'day' , y = 'total_bill',data= df1)
sns.pairplot(df1)
df.scatter_matrix()
sns.jointplot(x =df1.total_bill , y = df1.tip )
sns.jointplot(x =df1.total_bill , y = df1.tip,kind = 'hex' )
sns.regplot(x = df1.total_bill , y = df1.tip)
sns.set(rc={'figure.figsize':(20,10)})
```

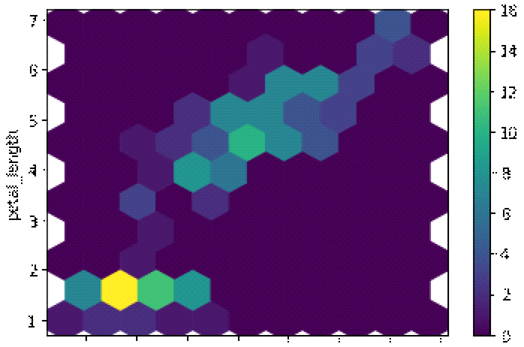
Plot.Scatter Graph



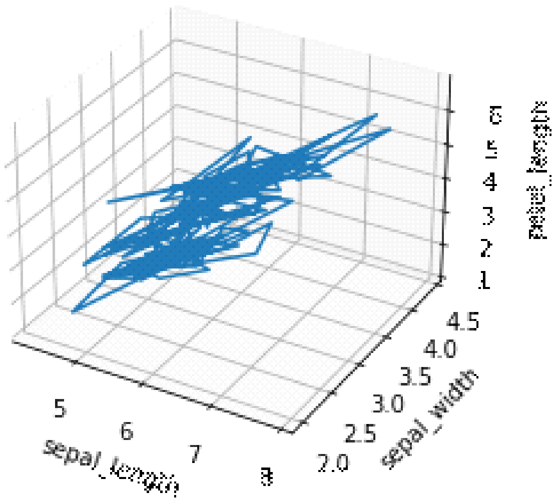




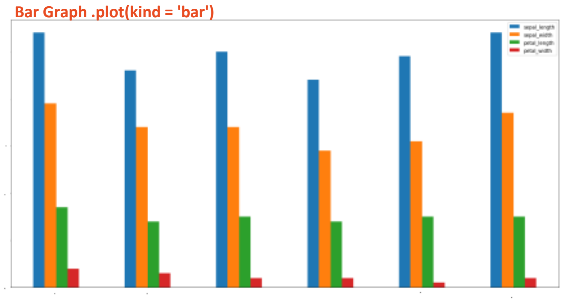
Plot.Hex-Bin



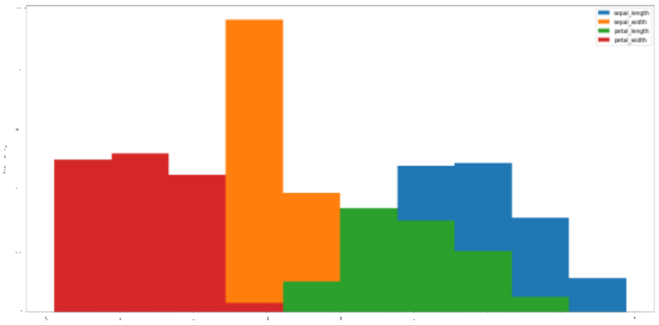
3-D Graph(plt.axes(projection='3d'))



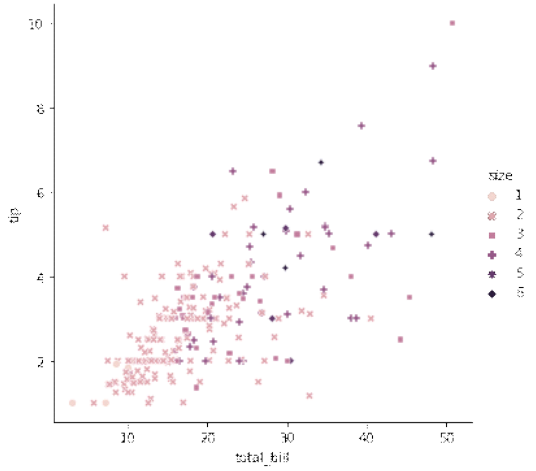
Bar Graph .plot(kind = 'bar')



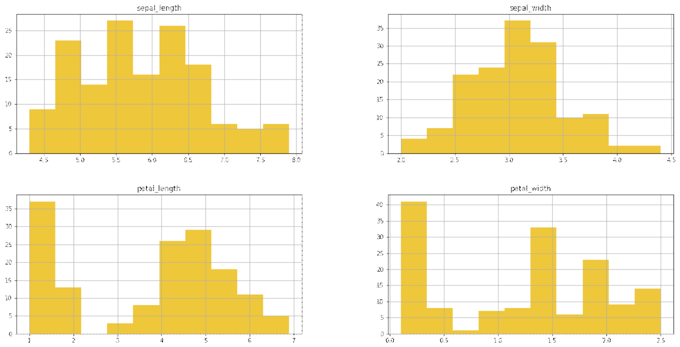
Histogram(.plot(kind='hist))



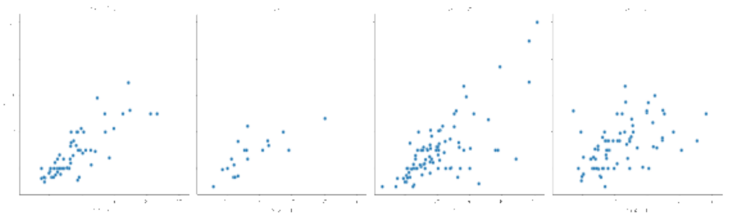
Sns.relplot()



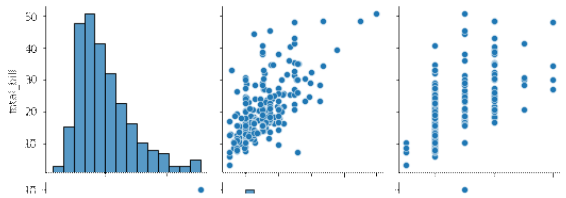
Histogram of Data (df.hist())



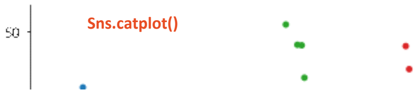
Sns.relplot()

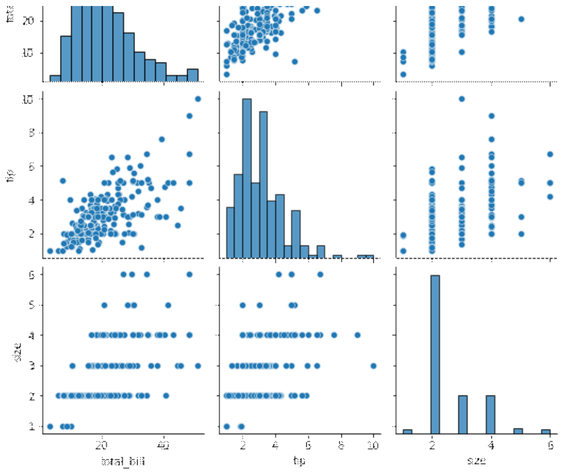


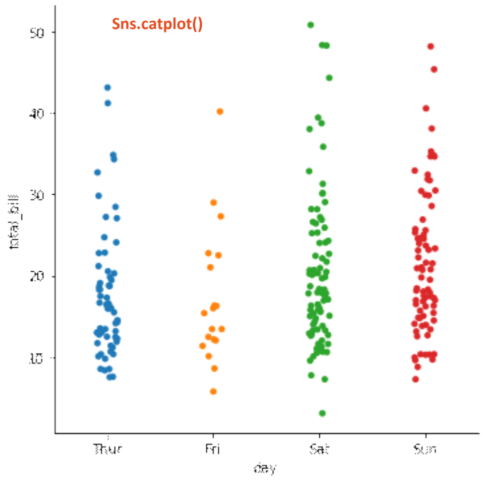
Sns.pairplot()

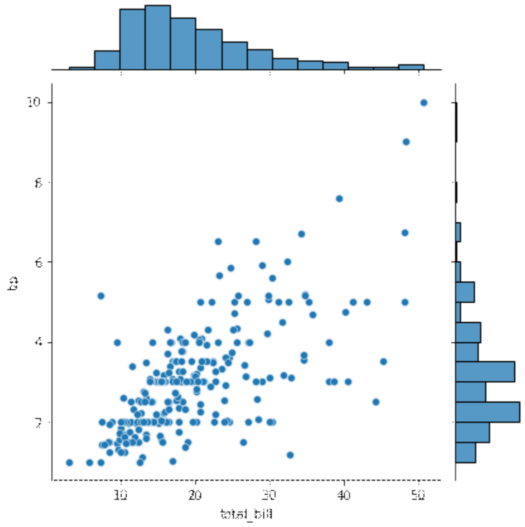


Sns.catplot()

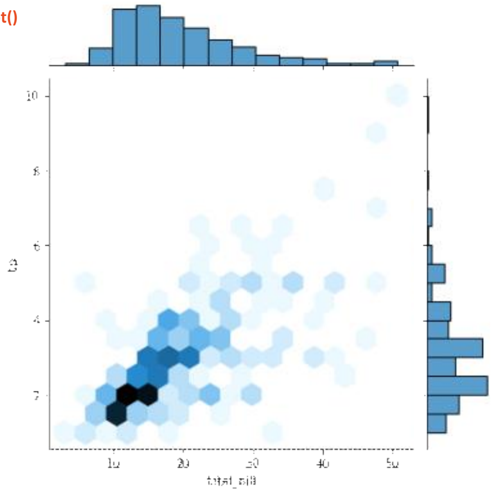





Sns.catplot()



Sns.jointplot()


tip
total_bill