Name: Ayush Chanchal

Sap id: 500097569

Batch: B6

Semester: 6

Assignment-1

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

# Loading Iris Dataset

df=pd.read\_csv("iris.csv")

# Displaying first 5 rows

df.head(5)

# Checking missing values

df.isnull().sum() # (In this Dataset there is no missing value)

# Summary of Dataset

df.describe()  # BY default take numerical column only

# Selecting subset of column using label based indexing

label\_based\_df=df[["petal\_length","sepal\_length","species"]]

# Selecting subset of column using position based indexing

index\_based\_df=df.iloc[:,[0,2,4]]

# Creating new dataframe by filtering rows

new\_df=df[df["species"]=="Iris-virginica"]

# Task-2

df=pd.read\_csv("iris.csv")

# Checking missing values

df.isnull().sum() # (In this Dataset there is no missing value)

# Creating a new column by applying mathematical operation

df["Total\_length"]=df["sepal\_length"]+df["petal\_length"]

# Converting cateogrical column into numerical Representation

one\_hot\_encoded\_data = pd.get\_dummies(df, *columns* = ["species"])

# grouping dataset by specific column and applying aggregate function

df\_mean=df.groupby("species").mean()

df\_count=df.groupby("species").count()

df\_sum=df.groupby("species").sum()

# Represnting in meaninful way

df\_mean.rename(*columns*={"petal\_length":"petal\_length\_mean","petal\_width":"petal\_width\_mean","sepal\_length":"sepal\_length\_mean","sepal\_width":"sepal\_width\_mean","Total\_length":"Total\_length\_mean"})

df\_count.rename(*columns*={"petal\_length":"petal\_length\_count","petal\_width":"petal\_width\_count","sepal\_length":"sepal\_length\_count","sepal\_width":"sepal\_width\_count","Total\_length":"Total\_length\_count"})

df\_sum.rename(*columns*={"petal\_length":"petal\_length\_sum","petal\_width":"petal\_width\_sum","sepal\_length":"sepal\_length\_sum","sepal\_width":"sepal\_width\_sum","Total\_length":"Total\_length\_sum"})

# Task-3

# Merge two different dataset using various type of join

df1=pd.read\_csv("2017.csv")

df2=pd.read\_csv("gapminder\_full.csv")

df2.rename(*columns*={"country":"Country"},*inplace*=True)

merged\_inner=pd.merge(df1,df2,*on*="Country",*how*="inner")

merged\_left=pd.merge(df1,df2,*on*="Country",*how*="left")

merged\_right=pd.merge(df1,df2,*on*="Country",*how*="right")

merged\_outer=pd.merge(df1,df2,*on*="Country",*how*="outer")

# Impact of each type of join

# Inner Join: Only the rows with matching country values in both datasets are included in the result. Rows with non-matching country values are excluded.

# Left Join: All rows from the left dataset (df1) are included, and matching rows from the right dataset (df2) are added. If there is no match in the right dataset, NaN values are filled.

# Right Join: All rows from the right dataset (df2) are included, and matching rows from the left dataset (df1) are added. If there is no match in the left dataset, NaN values are filled.

# Outer Join: All rows from both datasets are included. If there is a match, values are filled; otherwise, NaN values are used.

# Task-4

tips = sns.load\_dataset("tips")

sns.barplot(*x*="day", *y*="total\_bill", *data*=tips,*ci*=None)

plt.title('Bar Plot - Total Bill Amount by Day')

plt.show()

sns.lineplot(*x*="time", *y*="total\_bill", *data*=tips,*ci*=None)

plt.title('Line Plot - Total Bill Amount by Time')

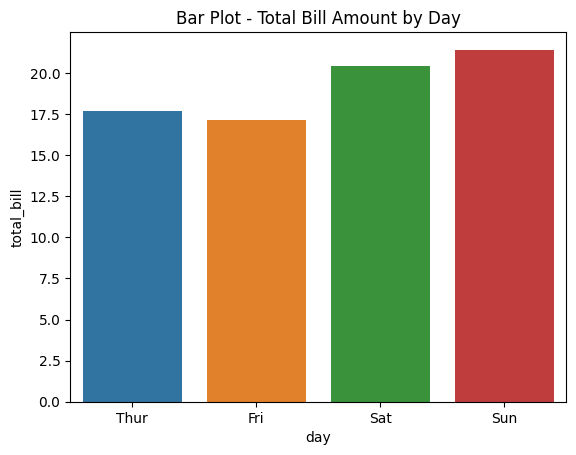
plt.show()

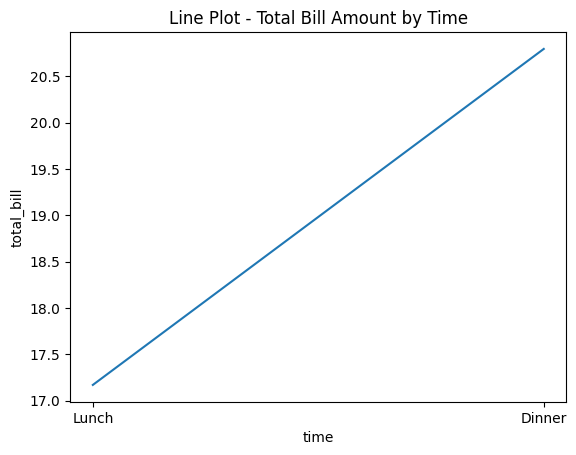
sns.scatterplot(*x*="total\_bill", *y*="tip", *data*=tips)

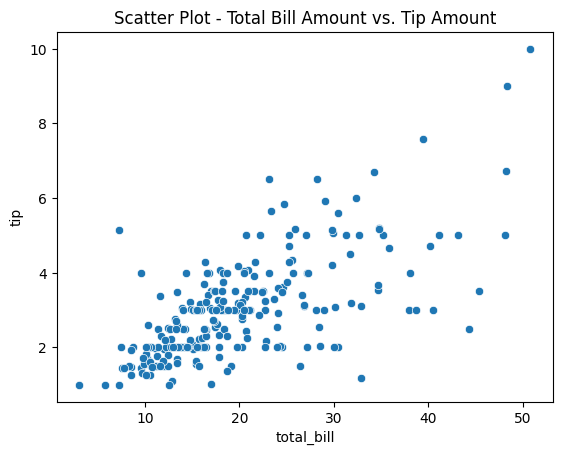
plt.title('Scatter Plot - Total Bill Amount vs. Tip Amount')

plt.show()

tips.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 |

correlation\_matrix = tips[["total\_bill","tip","size"]].corr()

# Ploting the correlation matrix using a heatmap

sns.heatmap(correlation\_matrix, *annot*=True, *cmap*="coolwarm", *fmt*=".2f", *linewidths*=0.5)

plt.title('Correlation Matrix')

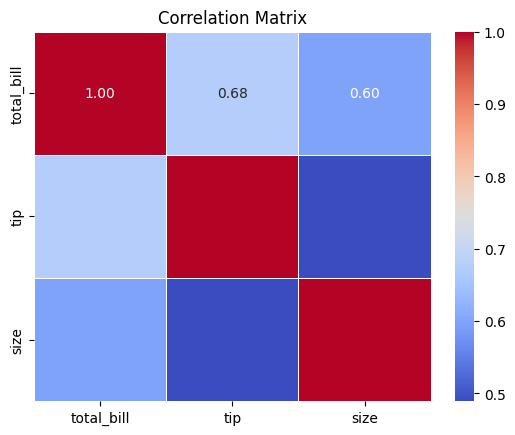
plt.show()

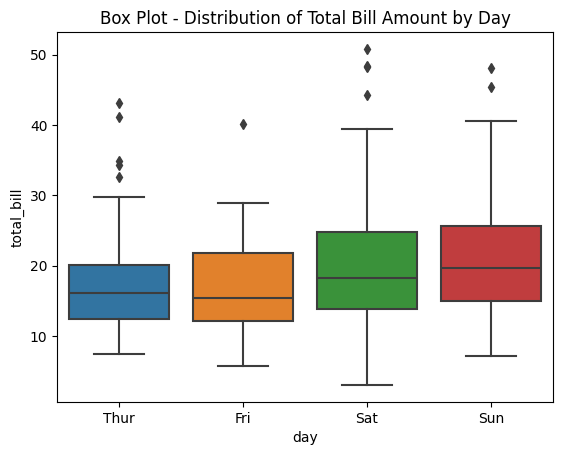
sns.boxplot(*x*="day", *y*="total\_bill", *data*=tips)

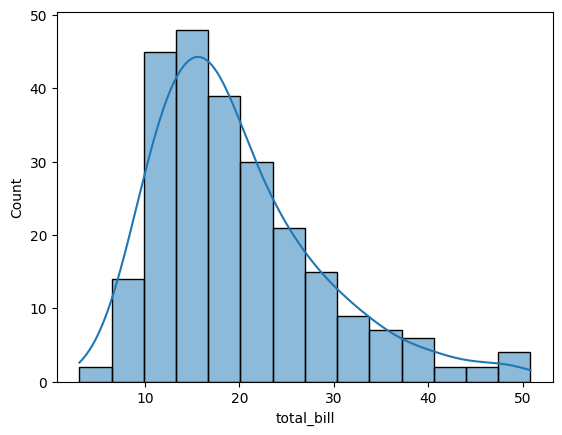
plt.title('Box Plot - Distribution of Total Bill Amount by Day')

plt.show()

sns.histplot(tips['total\_bill'], *kde*=True)







# Task-5

# Creating a numpy array

arr=np.array([1,2,3,4,5,6,7,8,9,10])

arr2=np.array([11,12,13,14,15,16,17,18,19,20])

# Performing Add,subtract,multiply,and divide on these arrays

add\_arr=np.add(arr,arr2)

print(add\_arr)

subtract\_arr=np.subtract(arr2,arr)

print(subtract\_arr)

multiply\_arr=np.multiply(arr,arr2)

print(multiply\_arr)

divide\_arr=np.divide(arr2,arr)

print(divide\_arr)

[12 14 16 18 20 22 24 26 28 30]

[10 10 10 10 10 10 10 10 10 10]

[ 11 24 39 56 75 96 119 144 171 200]

[11. 6. 4.33333333 3.5 3. 2.66666667

2.42857143 2.25 2.11111111 2. ]

# Task-6

# Reshaping array into (2,5)

res\_arr=np.reshape(arr,(2,5))

# Transpose Matrix

trans\_arr=np.transpose(res\_arr)

# Flattening the transposed matrix in 1D array

flatten\_arr=trans\_arr.flatten()

# Stacking arr ,arr2 vertically

stacked\_arr=np.vstack((arr,arr2))  # array dimension/shape should be same

print(stacked\_arr)

[[ 1 2 3 4 5 6 7 8 9 10]

[11 12 13 14 15 16 17 18 19 20]]

# Task-7

# Calculating mean ,median,standard deviation

mean\_of\_arr=arr.mean()

median\_of\_arr=np.median(arr)

standard\_dev\_of\_array=np.std(arr)

# Finding max and min of arr

max\_arr=arr.max()

min\_arr=arr.min()

# Normalising array

normalised\_arr=(arr-mean\_of\_arr)/standard\_dev\_of\_array

# Task-8

# Creating bool arr for element greater than 5

bool\_arr=arr>5

#  Using  boolean indexing to extract elements greater than 5

arr\_greater=arr[bool\_arr]

# Task-9

# Generate a 3x3 matrix with random values between 0 and 1.

arr3=np.random.rand(3,3)

# Create an array of 10 random integers between 1 and 100.

rand\_int\_arr=np.random.randint(1,100,*size*=10)

#Shuffle the elements of 'arr' randomly.

np.random.shuffle(arr)  #Keep in mind that np.random.shuffle modifies the input array in-place

arr

array([ 2, 8, 5, 10, 6, 1, 7, 9, 3, 4])

# Task-10

# Calculating sqrt of each element in arr

square\_root\_arr=np.sqrt(arr)

# Calculating exponent of each element of array

exp\_arr=np.exp(arr)

# Task-11

# Create a 3x3 matrix 'mat\_a' with random values.

mat\_a=np.random.rand(3,3)

# Create a 3x1 matrix 'vec\_b' with random values.

vec\_b=np.random.rand(3,1)

# .Multiply 'mat\_a' and 'vec\_b' using the dot product.

dot\_arr=np.dot(mat\_a,vec\_b)

# Task-12

matrix = np.array([[1, 2, 3],

                   [4, 5, 6],

                   [7, 8, 9]])

mean\_subtracted\_matrix = matrix - np.mean(matrix, *axis*=1, *keepdims*=True)

mean\_subtracted\_matrix

array([[-1., 0., 1.],

[-1., 0., 1.],

[-1., 0., 1.]])