**DATA SCIENCE LAB MANUAL**

1. **Consider the following data of three cricket players in 10 innings T20 Match**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Player** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **Cricketer1** | **25** | **10** | **55** | **45** | **55** | **78** | **55** | **0** | **49** | **10** |
| **Cricketer2** | **47** | **62** | **78** | **45** | **100** | **20** | **100** | **0** | **80** | **10** |
| **Cricketer3** | **80** | **17** | **7** | **10** | **45** | **79** | **75** | **75** | **80** | **42** |

1. **Find Whose average is better.**
2. **What is the middlemost value of each player?**
3. **Whose most frequent value is good.**
4. **Draw a simple plot to show performance of players.**

Solution:

#Cricket Player Performance Analysis  
import statistics as st  
import matplotlib.pyplot as pt  
import tabulate  
Matches=[1,2,3,4,5,6,7,8,9,10]  
Player1=[25,10,55,45,55,78,55,0,49,10]  
Player2=[47,62,78,45,100,20,100,0,80,10]  
Player3=[80,17,7,10,45,79,75,75,80,42]  
#Player1 Summary  
print("Player1 Mean = ",st.mean(Player1))  
print("Player1 Median = ",st.median(Player1))  
print("Player1 Mode = ",st.mode(Player1))  
#Player2 Summary  
print("Player2 Mean = ",st.mean(Player2))  
print("Player2 Median = ",st.median(Player2))  
print("Player2 Mode = ",st.mode(Player2))  
#Player3 Summary  
print("Player3 Mean = ",st.mean(Player3))  
print("Player3 Median = ",st.median(Player3))  
print("Player3 Mode = ",st.mode(Player3))  
#Performance plot  
pt.plot(Matches,Player1)  
pt.plot(Matches,Player2)  
pt.plot(Matches,Player3)  
pt.title("Cricket Player Performance")  
pt.xlabel("Matches")  
pt.ylabel("Scores")  
pt.legend(["Player1","Player2","Player3"])  
pt.show()

OUTPUT:

Player1 Mean = 38.2

Player1 Median = 47.0

Player1 Mode = 55

Player2 Mean = 54.2

Player2 Median = 54.5

Player2 Mode = 100

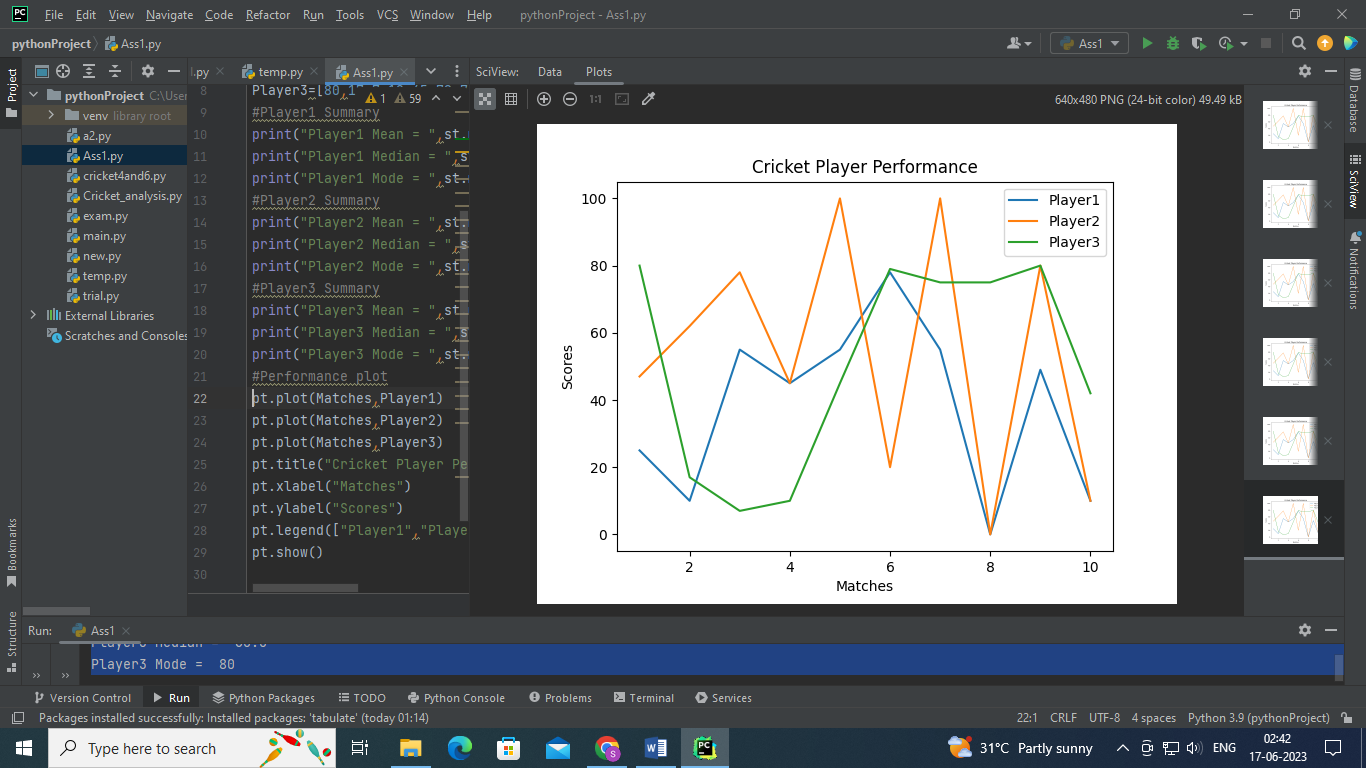
Player3 Mean = 51

Player3 Median = 60.0

Player3 Mode = 80

Analysis

1. Player 2 average is better.
2. Player1 Median = 47.0, Player2 Median = 54.5, Player3 Median = 60.0
3. Player2
4. Draw a simple plot to show performance of players.



1. Consider Insurance Dataset and analyze following
2. Count Number of Male and Female
3. What is average age of peoples.
4. Display simple bar plot Gender wise

Solution:

import pandas as pd  
import openpyxl  
import statistics as st  
import matplotlib.pyplot as pt  
data = pd.read\_csv("E:\Data Science with Python\DataSet\insurance.csv")  
print(data)  
#Analysis genderwise  
ls=data['sex'].tolist()  
y1=ls.count('female')  
y2=ls.count('male')  
print("female Count = ",y1)  
print("male Count = ",y2)  
  
#Aveage age of customers  
avgage=data['age'].tolist()  
print("Average Age= %.2f " % st.mean(avgage))  
  
#Display Histogram genderwise  
x=["FEMALE","MALE"]  
y=[y1,y2]  
pt.bar(x,y)  
pt.title("Genderwise Insurance Data")  
pt.xlabel("Gender")  
pt.ylabel("Count")  
pt.show()

Analysis:

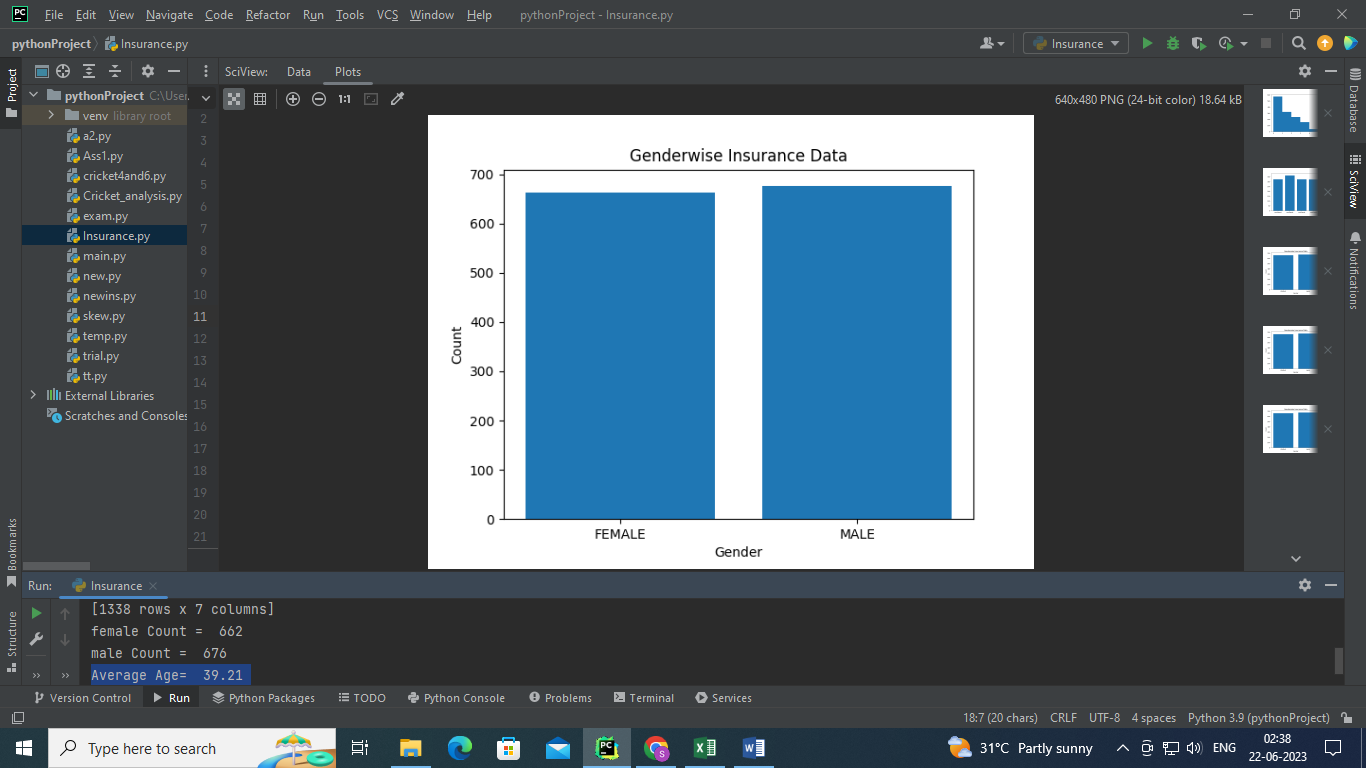
a)

female Count = 662

male Count = 676

b) Average Age= 39.21

c)



1. **Consider Insurance Dataset and analyze data region wise. Also display simple bar chart region wise.**

**Solution:**

import pandas as pd  
import openpyxl  
import matplotlib.pyplot as pt  
data = pd.read\_csv("E:\Data Science with Python\DataSet\insurance.csv")  
print(data)  
  
#Regionwise count  
region=data['region'].tolist()  
output=[]  
for x in region:  
 if x not in output:  
 output.append(x)  
print(output)  
y1=region.count('southwest')  
y2=region.count('southeast')  
y3=region.count('northwest')  
y4=region.count('northeast')  
print("Southwest count= ",y1)  
print("southeast count= ",y2)  
print("northwest count= ",y3)  
print("northeast count= ",y4)  
pt.title("Regionwise Count")  
pt.xlabel("Region")  
pt.ylabel("Count")  
y=[y1,y2,y3,y4]  
pt.bar(output,y)  
pt.show()

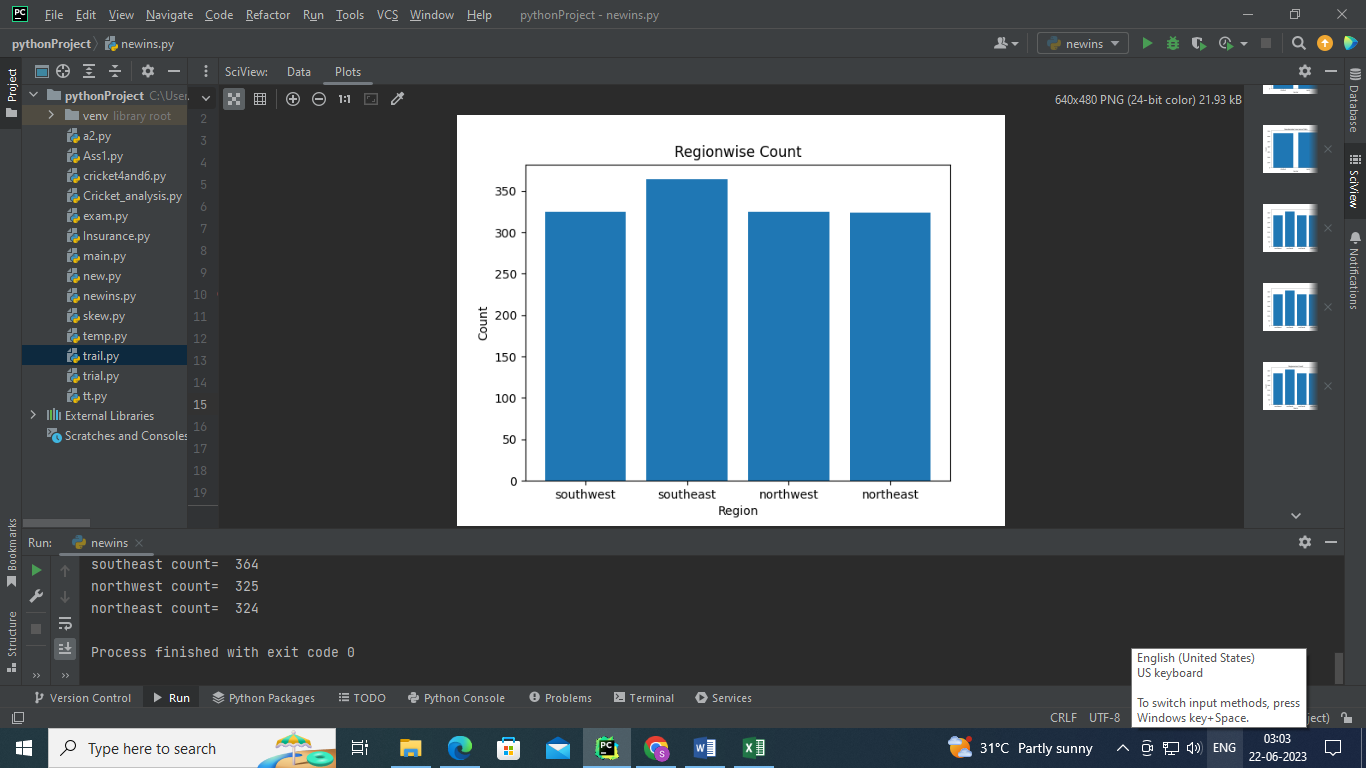
**Analysis:**

Southwest count= 325

southeast count= 364

northwest count= 325

northeast count= 324



1. Consider temperature dataset and analyze average of minimum and maximum temperature, minimum temperature, maximum temperature month wise.

Solution:

import pandas as pd  
import openpyxl  
import numpy as np  
data=pd.read\_excel("E:\\Data Science with Python\\DataSet\\belgavitemp2022.xlsx")  
print(data)  
df1 = (data.groupby(["Year", "Month"],sort=False).agg(Avg\_of\_Max\_Temp=("Max", 'mean'),  
 Max\_temp=("Max",'max'),Avg\_of\_Min\_Temp=("Min", 'mean'),Min\_temp=("Min",'min')))  
print(df1)

Analysis:

Avg\_of\_Max\_Temp Max\_temp Avg\_of\_Min\_Temp Min\_temp

Year Month

2022 January 29.290323 33 14.838710 11

February 32.535714 35 16.928571 14

March 35.451613 39 20.322581 17

April 36.666667 39 22.300000 19

May 33.838710 38 21.612903 19

June 31.533333 36 21.033333 20

July 28.225806 33 20.451613 19

August 28.419355 32 20.258065 19

September 29.533333 32 19.833333 18

October 29.741935 32 18.677419 14

November 30.433333 32 16.433333 11

December 29.870968 33 17.967742 14

**5.Consider following data and calculate Descriptive statistics using formules.**

22,26,14,30,18,1135,41,12,32

Solution:

import numpy as np  
import pandas as pd  
data=[22,26,14,30,18,11,35,41,12,32]  
print("Mean = %.2f"% np.mean(data))  
print("Median = ",np.median(data))  
print("Max = ",np.max(data))  
print("Min = ",np.min(data))  
print("First Quartile =",np.quantile(data,0.25))  
print("Second Quartile = ",np.quantile(data,0.50))  
print("Third Quartile = ",np.quantile(data,0.75))  
print("20 th Percentilee = ",np.percentile(data,20))  
print("99 th Percentilee = ",np.percentile(data,99))  
print("Standard deviation = %.2f" % np.std(data))  
print("Variance = ",np.var(data))

OUTPUT:

Mean = 24.10

Median = 24.0

Max = 41

Min = 11

First Quartile = 15.0

Second Quartile = 24.0

Third Quartile = 31.5

20 th Percentilee = 13.6

99 th Percentilee = 40.46

Standard deviation = 9.83

Variance = 96.69

1. Find the Quartiles for the following Students Score data and visualize graphically.

50,50,47,97,49,3,53,42,26,74,82,62,37,15,70,27,36,35,48,52,63,64.

Solution:

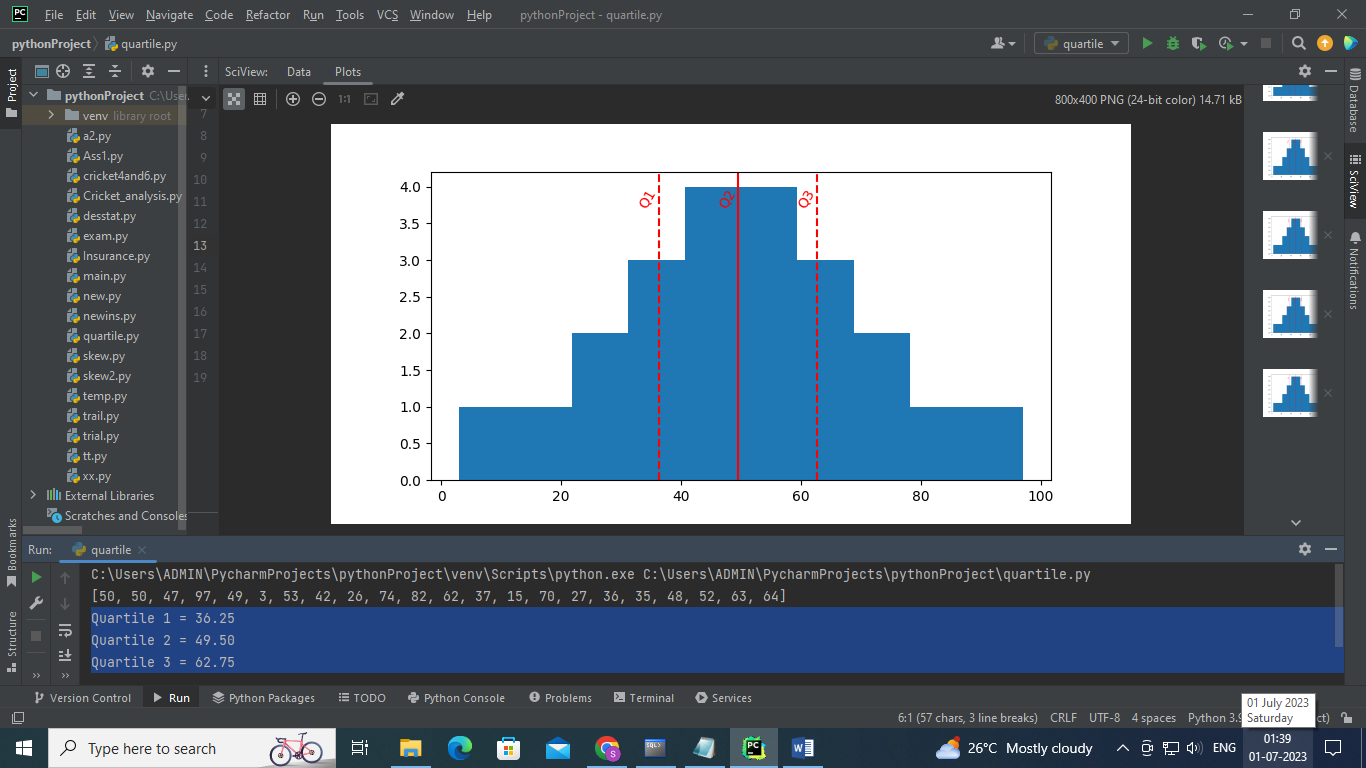
import numpy as np  
import matplotlib.pyplot as pt  
import numpy as np  
import pandas as pd  
data=[50,50,47,97,49,3,53,42,26,74,82,62,37,15,70,27,36,35,48,52,63,64]  
print(data)  
print("Quartile 1 = %.2f"%np.quantile(data,0.25))  
print("Quartile 2 = %.2f"%np.quantile(data,0.50))  
print("Quartile 3 = %.2f"%np.quantile(data,0.75))  
pt.figure(figsize=(8,4))  
pt.hist(data)  
# Vertical lines for each percentile of interest  
pt.axvline(np.quantile(data, 0.25), linestyle='--', color='red')  
pt.text(np.quantile(data, 0.25), 4, 'Q1', color='r', ha='right', va='top', rotation=60)  
pt.axvline(np.quantile(data, 0.50), linestyle='-', color='red')  
pt.text(np.quantile(data, 0.50), 4, 'Q2', color='r', ha='right', va='top', rotation=60)  
pt.axvline(np.quantile(data, 0.75), linestyle='--', color='red')  
pt.text(np.quantile(data, 0.75), 4, 'Q3', color='r', ha='right', va='top', rotation=60)  
pt.show()

OUTPUT:

Quartile 1 = 36.25

Quartile 2 = 49.50

Quartile 3 = 62.75



1. Calculate the skewness for the following data also conclude skewness

85,96,76,108,84,100,86,70,95,84

Solution

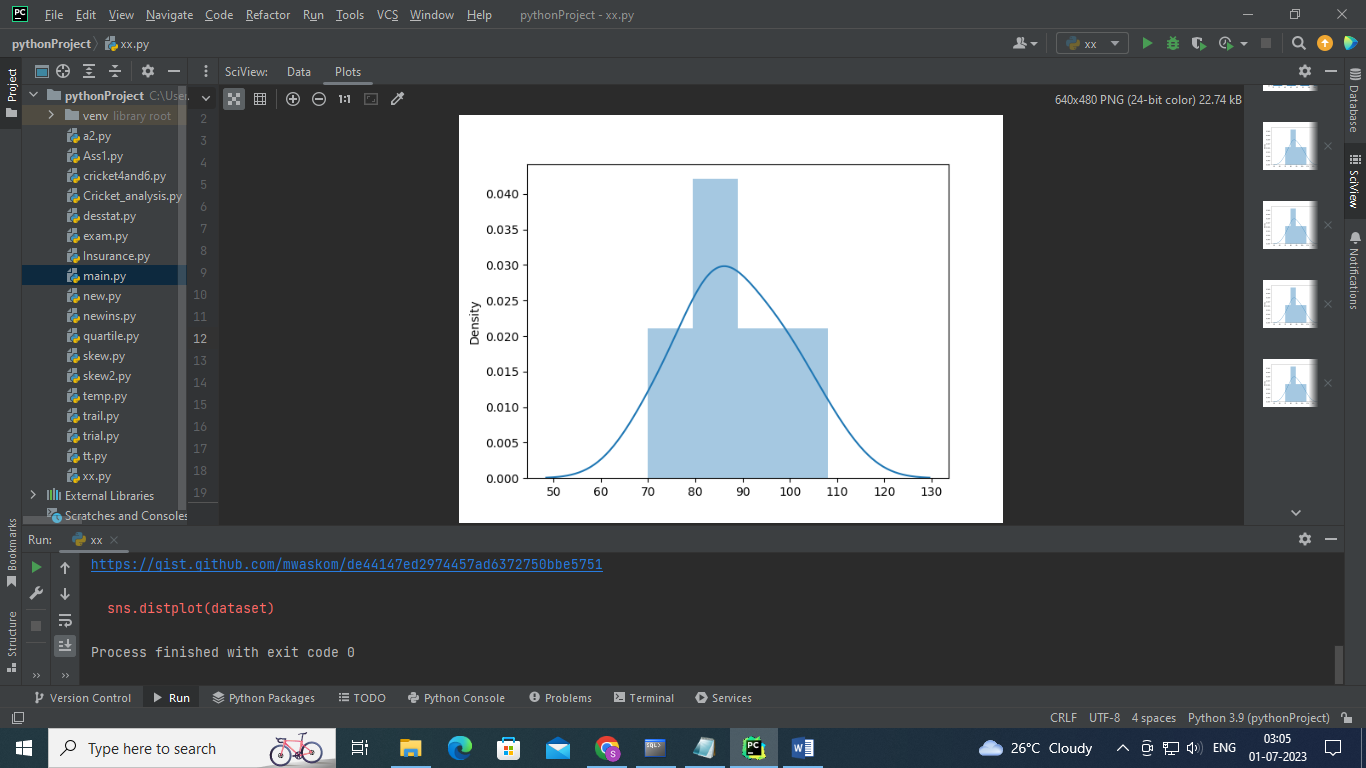
# Importing library  
import matplotlib.pyplot as pt  
import statistics as st  
import seaborn as sns  
# Creating a dataset  
dataset =[85,96,76,108,84,100,86,70,95,84]  
meandata=st.mean(dataset)  
print("Mean = %.2f"%meandata)  
modedata=st.mode(dataset)  
print("Mode = %.2f"%modedata)  
meddata=st.median(dataset)  
print("Median = %.2f"%meddata)  
# Calculate the skewness  
stddata=st.stdev(dataset)  
print("Standard Deviation =%.2f" % stddata)  
sk=(meandata-modedata)/stddata  
print("Skewness= %.2f" % sk)  
sns.distplot(dataset)  
pt.show()

OUTPUT:

Mean = 88.40

Mode = 84.00

Median = 85.50



Analysis: Distribution is Positively Skewed.

1. Consider Student Performance dataset and find skewness for all subjects.

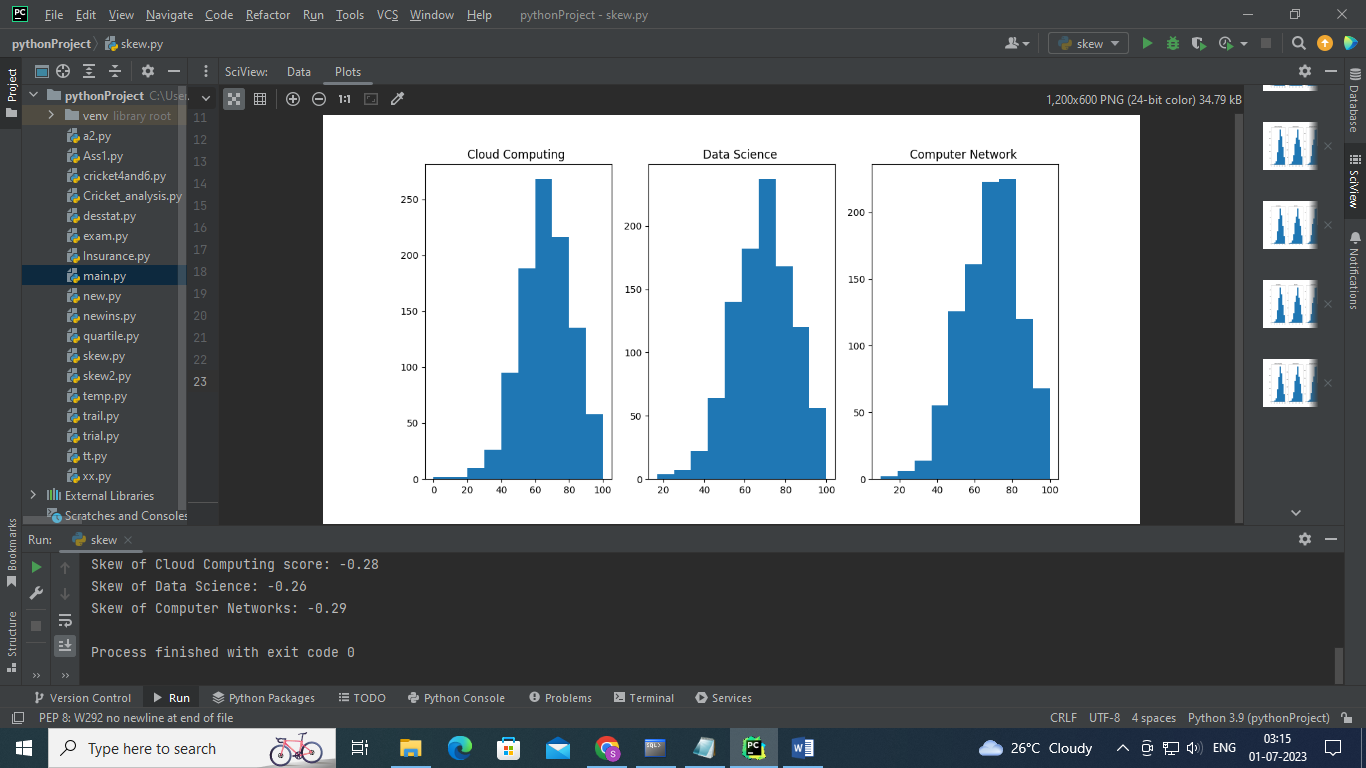
import pandas as pd  
import matplotlib.pyplot as plt  
import openpyxl  
data =pd.read\_csv("E:\Data Science with Python\DataSet\StudentsPerformance.csv")  
print(data)  
print("Skew of Cloud Computing score: %.2f"%data['Cloud Computing'].skew())  
print("Skew of Data Science: %.2f"%data['Data Science'].skew())  
print("Skew of Computer Networks: %.2f"%data['Computer Network'].skew())  
  
plt.figure(figsize = (12,6))  
plt.subplot(1, 3, 1)  
plt.hist(data['Cloud Computing'])  
plt.title('Cloud Computing ')  
  
plt.subplot(1, 3, 2)  
plt.hist(data['Data Science'])  
plt.title('Data Science ')  
  
plt.subplot(1,3,3)  
plt.hist(data['Computer Network'])  
plt.title('Computer Network ')  
  
plt.show()

OUTPUT:

Skew of Cloud Computing score: -0.28

Skew of Data Science: -0.26

Skew of Computer Networks: -0.29



Analysis:

All subjects Distribution is negatively skewed.

Maximum students score between 60-100.

1. Consider Student Performance dataset find basic statistics of data science subject using pandas describe function, calculate skewness also visualize distribution.

Solution:

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from scipy.stats import skew, skewtest, norm  
import openpyxl  
data =pd.read\_csv("E:\Data Science with Python\DataSet\StudentsPerformance.csv")  
print(data)  
print(data['Data Science'].describe())  
print("Skewness= %.2f"%data['Data Science'].skew())  
sns.distplot(data['Data Science'], fit=norm, color="r")  
plt.show()

OUTPUT:

count 1000.000000

mean 69.169000

std 14.600192

min 17.000000

25% 59.000000

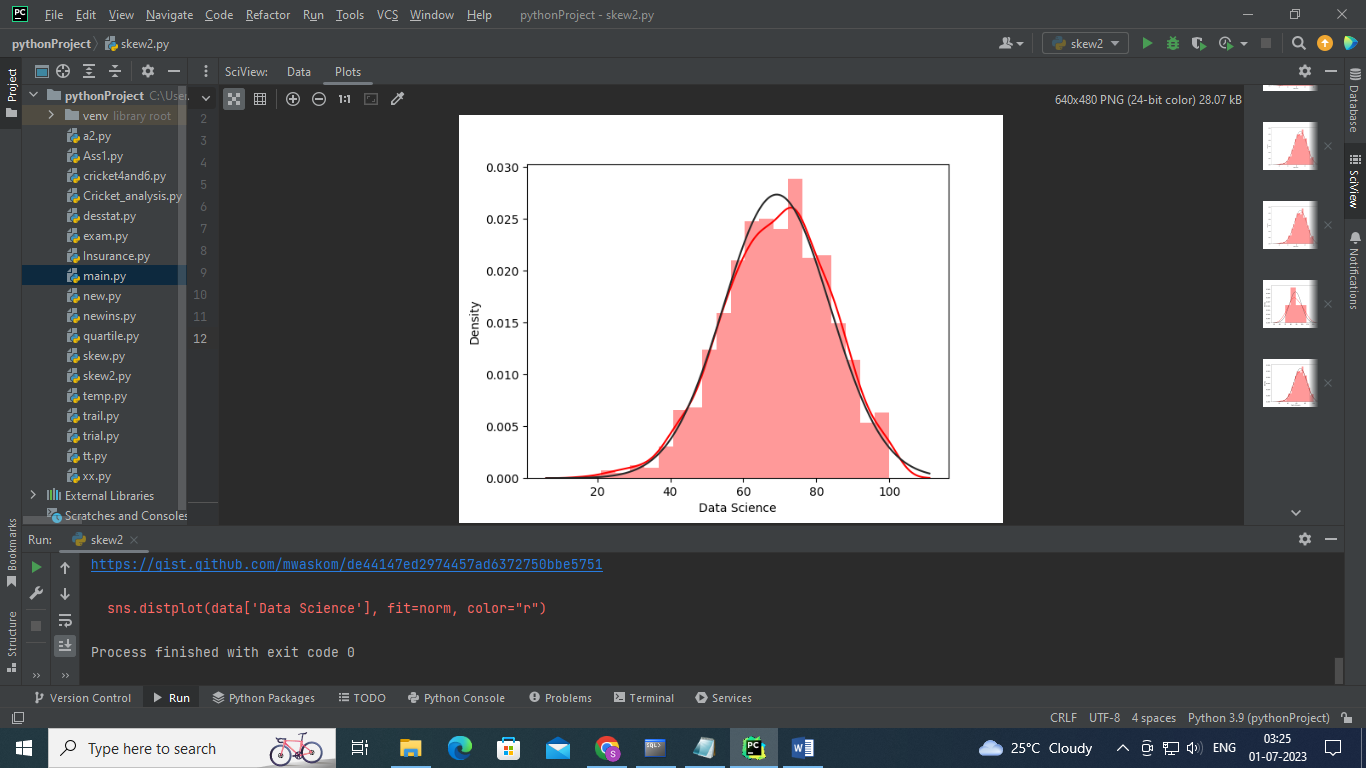
50% 70.000000

75% 79.000000

max 100.000000

Name: Data Science, dtype: float64

Skewness= -0.26



1. **Draw Regression Line for the following data. Conclude your analysis.**

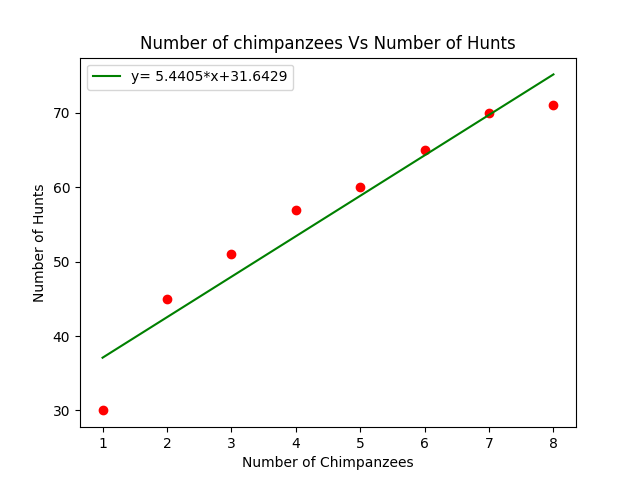
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. of chimpanzees | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| No. of hunting | 30 | 45 | 51 | 57 | 60 | 65 | 70 | 71 |

# Import packages  
import numpy as np  
import matplotlib.pyplot as plt  
x= np.array([1,2,3,4,5,6,7,8])  
# Dependent Variable - percent of successful hunts  
y = np.array([30,45,51,57,60,65,70,71])  
n = np.size(x)  
x\_mean = np.mean(x)  
y\_mean = np.mean(y)  
b1=n \* np.sum(x\*y)-np.sum(x)\*np.sum(y)  
b2=(n \* sum(x\*x) - (np.sum(x)\*np.sum(x)))  
b=(b1/b2)  
a= y\_mean-b\*x\_mean  
print("Line Slope is : %.4f"%b)  
print("Line Intercept is: %.4f"%a)  
y\_pred=b\*x+a  
plt.scatter(x, y, color = 'red')  
plt.plot(x, y\_pred, color = 'green',label='y= 5.4405\*x+31.6429')  
plt.xlabel('Number of Chimpanzees')  
plt.ylabel('Number of Hunts')  
plt.title("Number of chimpanzees Vs Number of Hunts")  
plt.legend()  
plt.show()

OUTPUT:

Line Slope is : 5.4405

Line Intercept is: 31.6429



**Analysis:**

Positive Correlation exist between number of chipanzees and number of hunts.

1. Consider Salary data and draw regression line using polyfit function and visualize graph. Conclude your analysis.

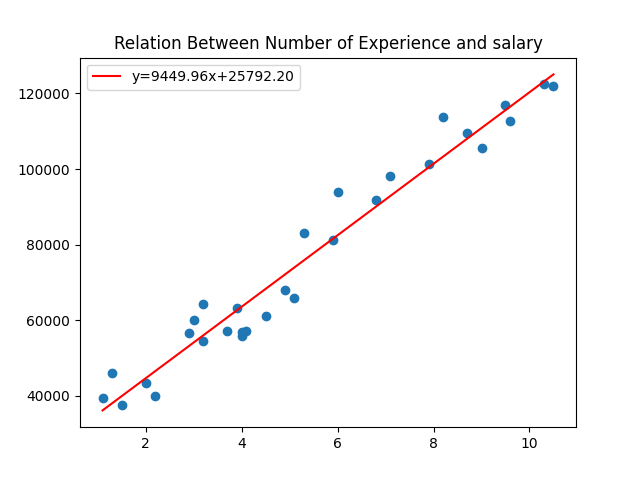
import pandas as pd  
import matplotlib.pyplot as plt  
import openpyxl  
import numpy as np  
data =pd.read\_csv("E:\Data Science with Python\DataSet\Salary\_Data.csv")  
print(data)  
x=data['YearsExperience']  
y=data['Salary']  
plt.plot(x, y, 'o')  
print("Correlation Coefficient = ",np.corrcoef(x,y))  
#obtain m (slope) and b(intercept) of linear regression line  
b, a = np.polyfit(x, y, 1)  
print("Slope= %.2f"%b,"Intercept = %.2f"%a)  
#add linear regression line to scatterplot  
plt.plot(x, b\*x+a,color='red',label='y=9449.96x+25792.20')  
plt.legend()  
plt.title("Relation Between Number of Experience and salary")  
plt.legend()  
plt.show()

OUTPUT:

Slope= 9449.96 Intercept = 25792.20

Correlation Coefficient = [[1. 0.97824162]

[0.97824162 1. ]]

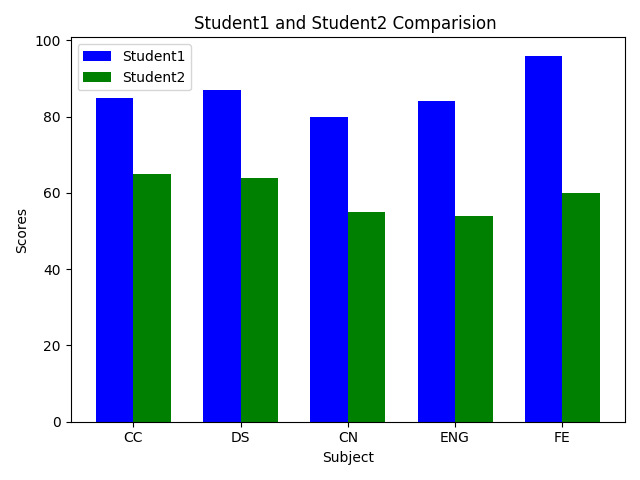


1. Display performance of two students in different subjects using bar graph. Also Comment on analysis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Student | CC | DS | ENG | CN | FE |
| Student1 | 85 | 87 | 80 | 84 | 96 |
| Student2 | 65 | 64 | 55 | 54 | 60 |

import matplotlib.pyplot as plt  
import numpy as np  
Stud1=[85,87,80,84,96]  
Stud2=[65,64,55,54,60]  
# create plot  
bar\_width = 0.35  
X = np.arange(5)  
p1 = plt.bar(X, Stud1, bar\_width, color='b',label='Student1')  
# The bar of second plot starts where the first bar ends  
p2 = plt.bar(X + bar\_width, Stud2, bar\_width,color='g',label='Student2')  
plt.xlabel('Subject')  
plt.ylabel('Scores')  
plt.title('Student1 and Student2 Comparision ')  
plt.xticks(X + (bar\_width/2) , ("CC","DS","CN","ENG","FE"))  
plt.legend()  
plt.tight\_layout()  
plt.show()

OUTPUT:



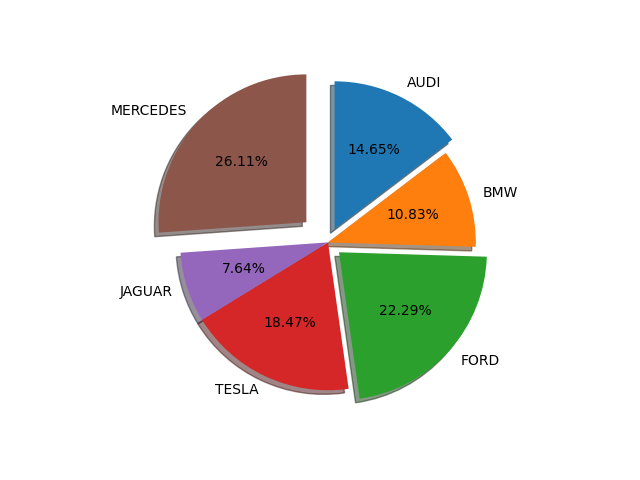
Student1 performance is good compare to student2.

1. Draw Pie chart for following data with explode, Shadow parameter.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| cars | AUDI | BMW | FORD | TESLA | JAGUAR | MERCEDES |
| data | 23 | 17 | 35 | 29 | 12 | 41 |

# Import libraries  
from matplotlib import pyplot as plt  
import numpy as np  
 # Creating dataset  
cars = ['AUDI', 'BMW', 'FORD','TESLA', 'JAGUAR', 'MERCEDES']  
data = [23, 17, 35, 29, 12, 41]  
 # Creating plot  
explode = [0.1, 0, 0.1, 0, 0,0.2]  
plt.pie(data, labels = cars,autopct='%1.2f%%',  
 explode=explode,shadow = True,startangle = 90,counterclock=False)  
 # show plot  
plt.show()

OUTPUT:

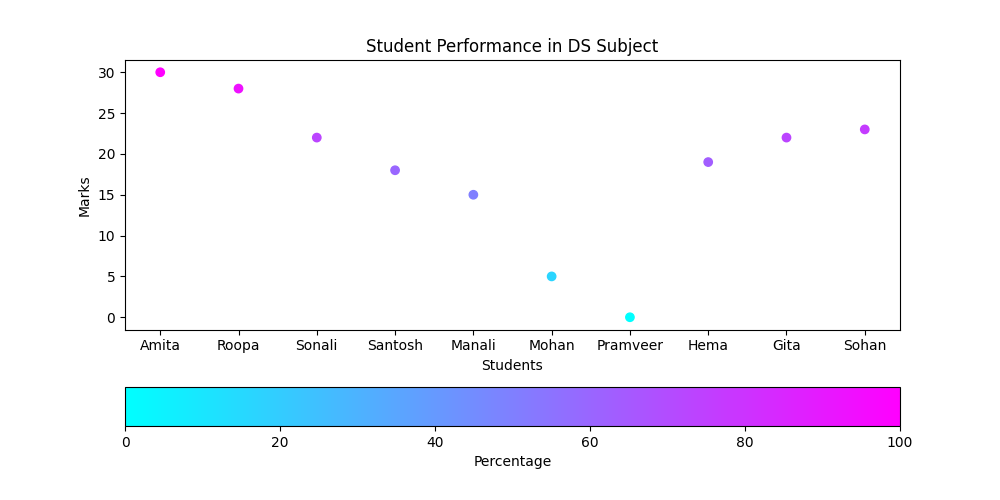


1. **Consider the following Marks data of students and draw color bar for percentage. Also analyze data. Given marks is out of 30.40% and above Passing percentage.**

marks= [30,28,22,18,15,5,0,19,22,23]

import matplotlib.pyplot as plt  
rollno= ["Amita","Roopa","Sonali","Santosh","Manali","Mohan","Pramveer","Hema","Gita","Sohan"]  
marks= [30,28,22,18,15,5,0,19,22,23]  
perls=[]  
for i in marks:  
 per="%.2f"%(i/30\*100)  
 perls.append(float(per))  
plt.figure(figsize=(10, 5))  
plt.scatter(x=rollno, y=marks, c=perls, cmap="cool")  
plt.colorbar(label="Percentage", orientation="horizontal")  
plt.title("Student Performance in DS Subject")  
plt.xlabel("Students")  
plt.ylabel("Marks")  
plt.show()

**OUTPUT:**

****

**Analysis:**

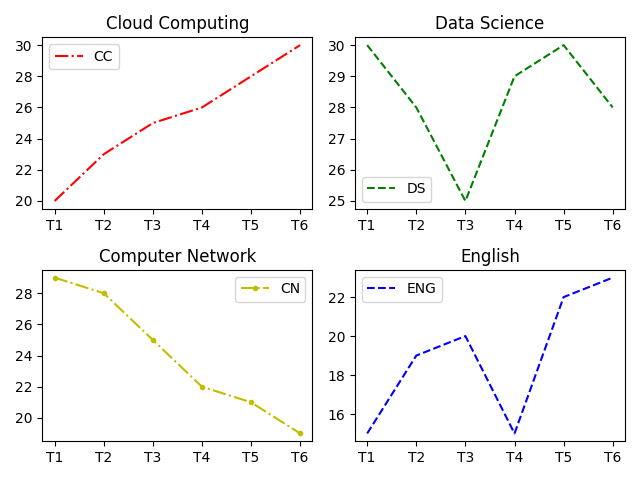
**Mohan and pramveer is failed because their percentage is between0 to 20. Remaining 6 students Passed DS exam.**

1. **Draw subplot 2 by 2 for the following data of student deepali in for different subjects. Comment your analysis.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test** | **T1** | **T2** | **T3** | **T4** | **T5** | **T6** |
| **CC** | **20** | **23** | **25** | **26** | **28** | **30** |
| **DS** | **30** | **28** | **25** | **29** | **30** | **28** |
| **CN** | **29** | **28** | **25** | **22** | **21** | **19** |
| **ENG** | **15** | **19** | **20** | **15** | **22** | **23** |

import matplotlib.pyplot as plt  
Test=['T1','T2','T3','T4','T5','T6']  
CC=[20,23,25,26,28,30]  
DS=[30,28,25,29,30,28]  
CN=[29,28,25,22,21,19]  
ENG=[15,19,20,15,22,23]  
plt.figure(figsize=(10,6))  
fig, ax = plt.subplots(2,2)  
ax[0,0].plot(Test,CC,'r-.',label='CC')  
ax[0,0].legend()  
ax[0,1].plot(Test,DS,'g--',label='DS')  
ax[0,1].legend()  
ax[1,0].plot(Test,CN,'y.-.',label='CN')  
ax[1,0].legend()  
ax[1,1].plot(Test,ENG,'b--',label='ENG')  
ax[1,1].legend()  
ax[0, 0].set\_title("Cloud Computing")  
ax[0, 1].set\_title("Data Science")  
ax[1, 0].set\_title("Computer Network")  
ax[1, 1].set\_title("English")  
# set spacing  
fig.tight\_layout()  
plt.show()

OUTPUT:



**Analysis:**

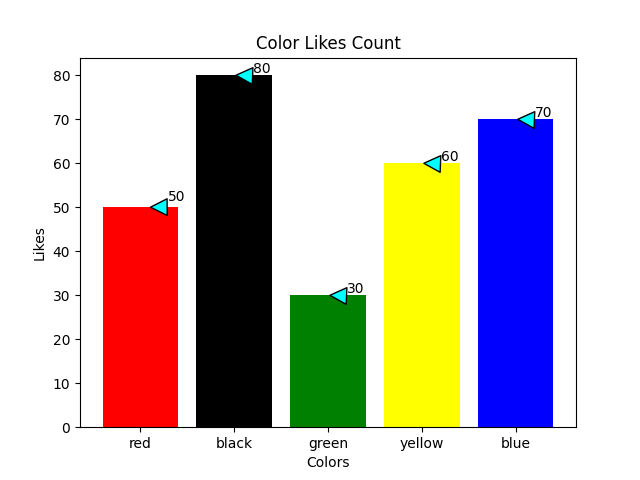
Cloud Computing performance increased whereas Computer network decreased.

1. Draw text Annotation for following data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Color | red | black | green | yellow | blue |
| Likes | 50 | 80 | 30 | 60 | 70 |

import matplotlib.pyplot as plt  
import numpy as np  
color=['red','black','green','yellow','blue']  
likes=[50,80,30,60,70]  
f, ax = plt.subplots()  
ax.bar(color,likes,color=color)  
ax.annotate(50, xy=(0.1, 50), xytext=(0.3, 51.5),  
 arrowprops=dict(facecolor='cyan', shrink=0.05,connectionstyle="angle3"))  
ax.annotate(80, xy=(1, 80), xytext=(1.2, 80.5),  
 arrowprops=dict(facecolor='cyan', shrink=0.1))  
ax.annotate(30, xy=(2, 30), xytext=(2.2, 30.5),  
 arrowprops=dict(facecolor='cyan', shrink=0.1))  
ax.annotate(60, xy=(3, 60), xytext=(3.2, 60.5),  
 arrowprops=dict(facecolor='cyan', shrink=0.1))  
ax.annotate(70, xy=(4, 70), xytext=(4.2, 70.5),  
 arrowprops=dict(facecolor='cyan', shrink=0.1))  
plt.title("Color Likes Count")  
plt.xlabel("Colors")  
plt.ylabel("Likes")  
plt.show()

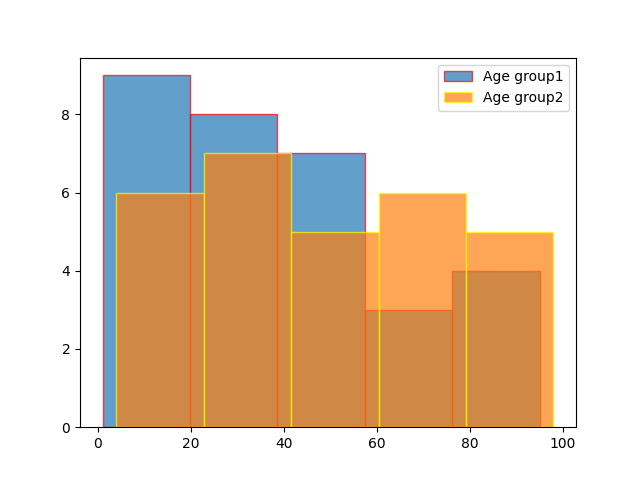
OUTPUT:



1. Display Histogram comparison for following data. Also comment your analysis.

# importing libraries  
import matplotlib.pyplot as plt  
# giving two age groups data  
age\_g1 = [1, 3, 5, 10, 15, 17, 18, 16, 19,  
 21, 23, 28, 30, 31, 33, 38, 32,  
 40, 45, 43, 49, 55, 53, 63, 66,  
 85, 80, 57, 75, 93, 95]  
  
age\_g2 = [6, 4, 15, 17, 19, 21, 28, 23, 31,  
 36, 39, 32, 50, 56, 59, 74, 79, 34,  
 98, 97, 95, 67, 69, 92, 45, 55, 77,  
 76, 85]  
# plotting first histogram  
plt.hist(age\_g1, label='Age group1', bins=5, alpha=.7, edgecolor='red')  
# plotting second histogram  
plt.hist(age\_g2, label="Age group2", bins=5, alpha=.7, edgecolor='yellow')  
plt.legend()  
# Showing the plot using plt.show()  
plt.show()

OUTPUT:



In Age group1 0-20 years’ people are more where as in age group2 60-80 peoples are more.

1. Display 2D ndarray basic operation accessing, inserting, deleting, updating elements operations also show additional functions of numpy array.

import numpy as np  
#Create 2 D Array  
arr=np.array([[1,2,3],[4,5,6],[7,8,9]])  
arr1=np.array([[10,11,12],[13,14,15],[16,17,18]])  
#print array  
print("Array = ",arr)  
#Display Dimesion of array  
print("Dimesion of array = ",arr.ndim)  
#Display Shape of Array  
print("Dimesion of array = ",arr.shape)  
# Access element 5  
print("Accessed Element= ",arr[1,1])  
#Insert new value at position 1 rowwise  
arr=np.insert(arr,1,[9,4,7],axis=0)  
print("After Insertion = ",arr)  
#Modification 8 with 88  
arr[3,1]=88  
print("After Modification = ",arr)  
#Deleting elemnts  
print(arr)  
arr = np.delete(arr, 1, axis=0)  
print("After Deletion = ",arr)  
#Addtional numpy array functions  
print("Transpose of matrix= ",np.transpose(arr))  
print("After Concatnation Columnwise of arr and arr1= ", np.concatenate((arr,arr1),axis=1))  
print("After Vetical stack operation on arr and arr1= ",np.vstack((arr,arr1)))  
print("After Horizontal stack operation on arr and arr1= ",np.hstack((arr,arr1)))

OUTPUT:

Array = [[1 2 3]

[4 5 6]

[7 8 9]]

Dimesion of array = 2

Dimesion of array = (3, 3)

Accessed Element= 5

After Insertion = [[1 2 3]

[9 4 7]

[4 5 6]

[7 8 9]]

After Modification = [[ 1 2 3]

[ 9 4 7]

[ 4 5 6]

[ 7 88 9]]

[[ 1 2 3]

[ 9 4 7]

[ 4 5 6]

[ 7 88 9]]

After Deletion = [[ 1 2 3]

[ 4 5 6]

[ 7 88 9]]

Transpose of matrix= [[ 1 4 7]

[ 2 5 88]

[ 3 6 9]]

After Concatnation Columnwise of arr and arr1= [[ 1 2 3 10 11 12]

[ 4 5 6 13 14 15]

[ 7 88 9 16 17 18]]

After Vetical stack operation on arr and arr1= [[ 1 2 3]

[ 4 5 6]

[ 7 88 9]

[10 11 12]

[13 14 15]

[16 17 18]]

After Horizontal stack operation on arr and arr1= [[ 1 2 3 10 11 12]

[ 4 5 6 13 14 15]

[ 7 88 9 16 17 18]]

1. Display 3D ndarray basic operation accessing, inserting, deleting, updating elements.

import numpy as np  
arr=np.array([[[1,2,3],[4,5,6]],  
 [[7,8,9],[10,11,12]],  
 [[13,14,15],[16,17,18]]])  
arr1=np.array([[[19,20,21],[22,23,24]],  
 [[25,26,27],[28,29,30]],  
 [[31,32,33],[34,35,36]]])  
#Print Dimnsion and shape  
print("Dimension= ",arr.ndim,"Shape = ",arr.shape)  
#Access 5  
print("Accessing Element 5 =",arr[0,1,1])  
#Access 10,11,12  
print("Accessing Element [10,11,12] =",arr[1,1,:])  
#Insert new row [[19,20,21],[22,23,24]]  
arr=np.insert(arr,3,[[19,20,21],[22,23,24]],axis=0)  
print("After Insertion",arr)  
#Modify 8 to 18  
arr[1,0,1]=18  
print("After Modifying 8 to 18 = ",arr)  
#Delete row 2  
arr=np.delete(arr,2,axis=0)  
print("After deleting 2 row = ",arr)  
#Addtional Functions  
print("Transpose of matrix= ",np.transpose(arr))  
print("After Concatnation Columnwise of arr and arr1= ", np.concatenate((arr,arr1),axis=1))  
print("After Vetical stack operation on arr and arr1= ",np.vstack((arr,arr1)))  
print("After Horizontal stack operation on arr and arr1= ",np.hstack((arr,arr1)))

OUTPUT:

Dimension= 3 Shape = (3, 2, 3)

Accessing Element 5 = 5

Accessing Element [10,11,12] = [10 11 12]

After Insertion [[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]

[[13 14 15]

[16 17 18]]

[[19 20 21]

[22 23 24]]]

After Modifying 8 to 18 = [[[ 1 2 3]

[ 4 5 6]]

[[ 7 18 9]

[10 11 12]]

[[13 14 15]

[16 17 18]]

[[19 20 21]

[22 23 24]]]

After deleting 2 row = [[[ 1 2 3]

[ 4 5 6]]

[[ 7 18 9]

[10 11 12]]

[[19 20 21]

[22 23 24]]]

Transpose of matrix= [[[ 1 7 19]

[ 4 10 22]]

[[ 2 18 20]

[ 5 11 23]]

[[ 3 9 21]

[ 6 12 24]]]

After Concatnation Columnwise of arr and arr1= [[[ 1 2 3]

[ 4 5 6]

[19 20 21]

[22 23 24]]

[[ 7 18 9]

[10 11 12]

[25 26 27]

[28 29 30]]

[[19 20 21]

[22 23 24]

[31 32 33]

[34 35 36]]]

After Vetical stack operation on arr and arr1= [[[ 1 2 3]

[ 4 5 6]]

[[ 7 18 9]

[10 11 12]]

[[19 20 21]

[22 23 24]]

[[19 20 21]

[22 23 24]]

[[25 26 27]

[28 29 30]]

[[31 32 33]

[34 35 36]]]

After Horizontal stack operation on arr and arr1= [[[ 1 2 3]

[ 4 5 6]

[19 20 21]

[22 23 24]]

[[ 7 18 9]

[10 11 12]

[25 26 27]

[28 29 30]]

[[19 20 21]

[22 23 24]

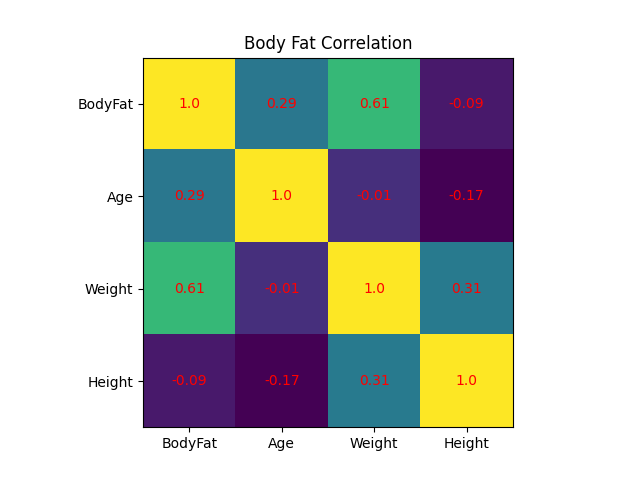
[31 32 33]

[34 35 36]]]

1. For bodyfat dataset calculate Correlation and Visualize Using Hitmap.

import matplotlib.pyplot as plt  
import pandas as pd  
import openpyxl  
import numpy as np  
import seaborn as sns  
data=pd.read\_csv("E:\\Data Science with Python\\DataSet\\bodyfat.csv")  
print(data)  
corr=data.corr()  
print(corr)  
fig ,ax=plt.subplots()  
plt.title("Body Fat Correlation")  
im= ax.imshow(corr.values)  
# set labels  
ax.set\_xticks(np.arange(len(corr.columns)))  
ax.set\_yticks(np.arange(len(corr.columns)))  
ax.set\_xticklabels(corr.columns)  
ax.set\_yticklabels(corr.columns)  
#Adding values  
for i in range(len(corr.columns)):  
 for j in range(len(corr.columns)):  
 text = ax.text(j, i, np.around(corr.iloc[i, j], decimals=2),  
 ha="center", va="center", color="red")  
plt.show()

OUTPUT:



1. Create dataframe in python for IPL Data and apply some basic operation on dataframe.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Team | MI | CSK | Devils | MI | CSK | RCB | CSK | CSK | KKR | KKR | KKR |
| Year | 2014 | 2015 | 2014 | 2015 | 2014 | 2015 | 2016 | 2017 | 2016 | 2014 | 2015 |
| Points | 876 | 789 | 863 | 673 | 741 | 812 | 756 | 788 | 694 | 701 | 804 |

import pandas as pd  
df=pd.DataFrame({"Team":["MI","CSK","Devils","MI","CSK","RCB","CSK",  
 "CSK","KKR","KKR","KKR"],  
 "Rank":[1,2,2,3,3,4,1,1,2,4,1],  
 "Year":[2014,2015,2014,2015,2014,2015,2016,2017,  
 2016,2014,2015],  
 "Points":[876,789,863,673,741,812,756,788,694,  
 701,804]},  
 index=["R1","R2","R3","R4","R5","R6","R7","R8",  
 "R9","R10","R11"])  
print("DataFrame = ")  
print(df)  
#Access Rows 2,4,6,8 using index and using labels  
print("After Accessing Rows 2,4,6,8 Using Labels = ")  
print(df.loc[["R2","R4","R6","R8"]])  
print("After Accessing Rows 2,4,6,8 Using Index = ")  
print(df.iloc[1:8:2])  
#Access top 3 Rows and also bottom 3 rowa  
print("Top 3 Rows = ")  
print(df.head(3))  
print("Bottom 3 Rows= ")  
print(df.tail(3))  
#Access columns team and points  
print("After Accessing 2 Columns Team and Points= ")  
print(df[['Team','Points']])  
#Access Row 3 and column 1,3,4 using index  
print("After Accessing row 3 and Columns 1,3,4 using index= ")  
print(df.iloc[2,[0,2,3]])  
#Access Row 3 and column 1,3,4 using labels  
print("After Accessing row 3 and Columns 1,3,4 using labels= ")  
print(df.loc["R3",['Team','Year','Points']])  
#Update last record with values 'RCB',3,2016,800  
df.iloc[10]=['RCB',3,2016,800]  
print("After Updating Last Row = ")  
print(df)  
#Insert new record in dataframe  
df.loc[len(df.index)] = ['MI',2,2017,800]  
print("After Inserting Last Row= ")  
print(df)  
#Delete row from dataframe  
df=df.drop([11])  
print("After Deleting Last Row = ")  
print(df)

OUTPUT: DataFrame =

Team Rank Year Points

R1 MI 1 2014 876

R2 CSK 2 2015 789

R3 Devils 2 2014 863

R4 MI 3 2015 673

R5 CSK 3 2014 741

R6 RCB 4 2015 812

R7 CSK 1 2016 756

R8 CSK 1 2017 788

R9 KKR 2 2016 694

R10 KKR 4 2014 701

R11 KKR 1 2015 804

After Accessing Rows 2,4,6,8 Using Labels =

Team Rank Year Points

R2 CSK 2 2015 789

R4 MI 3 2015 673

R6 RCB 4 2015 812

R8 CSK 1 2017 788

After Accessing Rows 2,4,6,8 Using Index =

Team Rank Year Points

R2 CSK 2 2015 789

R4 MI 3 2015 673

R6 RCB 4 2015 812

R8 CSK 1 2017 788

Top 3 Rows =

Team Rank Year Points

R1 MI 1 2014 876

R2 CSK 2 2015 789

R3 Devils 2 2014 863

Bottom 3 Rows=

Team Rank Year Points

R9 KKR 2 2016 694

R10 KKR 4 2014 701

R11 KKR 1 2015 804

After Accessing 2 Columns Team and Points=

Team Points

R1 MI 876

R2 CSK 789

R3 Devils 863

R4 MI 673

R5 CSK 741

R6 RCB 812

R7 CSK 756

R8 CSK 788

R9 KKR 694

R10 KKR 701

R11 KKR 804

After Accessing row 3 and Columns 1,3,4 using index=

Team Devils

Year 2014

Points 863

Name: R3, dtype: object

After Accessing row 3 and Columns 1,3,4 using labels=

Team Devils

Year 2014

Points 863

Name: R3, dtype: object

After Updating Last Row =

Team Rank Year Points

R1 MI 1 2014 876

R2 CSK 2 2015 789

R3 Devils 2 2014 863

R4 MI 3 2015 673

R5 CSK 3 2014 741

R6 RCB 4 2015 812

R7 CSK 1 2016 756

R8 CSK 1 2017 788

R9 KKR 2 2016 694

R10 KKR 4 2014 701

R11 RCB 3 2016 800

After Inserting Last Roe=

Team Rank Year Points

R1 MI 1 2014 876

R2 CSK 2 2015 789

R3 Devils 2 2014 863

R4 MI 3 2015 673

R5 CSK 3 2014 741

R6 RCB 4 2015 812

R7 CSK 1 2016 756

R8 CSK 1 2017 788

R9 KKR 2 2016 694

R10 KKR 4 2014 701

R11 RCB 3 2016 800

11 MI 2 2017 800

After Deleting Last Row =

Team Rank Year Points

R1 MI 1 2014 876

R2 CSK 2 2015 789

R3 Devils 2 2014 863

R4 MI 3 2015 673

R5 CSK 3 2014 741

R6 RCB 4 2015 812

R7 CSK 1 2016 756

R8 CSK 1 2017 788

R9 KKR 2 2016 694

R10 KKR 4 2014 701

R11 RCB 3 2016 800

1. **Create Data Frame for following data and analyze following**

* **Find Teams in year 2014**
* **Find Teams in whose Rank is 1**
* **Find Team with rank 2 and 3**
* **Find Teams 2014 or 2015**
* **Grouping on year and calculate mean of points**
* **Grouping on Team and calculate mean of points**
* **Maximum points in each year**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team** | **Rank** | **Year** | **Points** |
| **Riders** | **1** | **2014** | **876** |
| **Riders** | **2** | **2015** | **789** |
| **Devils** | **2** | **2014** | **863** |
| **Devils** | **3** | **2015** | **673** |
| **Kings** | **3** | **2014** | **741** |
| **Kings** | **4** | **2015** | **812** |
| **Kings** | **1** | **2016** | **756** |
| **Kings** | **1** | **2017** | **788** |
| **Riders** | **2** | **2016** | **694** |
| **Royals** | **4** | **2014** | **701** |
| **Royals** | **1** | **2015** | **804** |
| **Riders** | **2** | **2017** | **690** |

import pandas as pd  
ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',  
 'Kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],  
 'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],  
 'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],  
 'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}  
df = pd.DataFrame(ipl\_data)  
print("DataFrame = ")  
print(df)  
#Find Teams in year 2014  
print("Teams in year 2014 = ")  
print(df[df['Year'] == 2014])  
#Find Teams in whose Rank is 1  
print("Teams in whose Rank is 1 = ")  
print(df[df['Rank'] ==1])  
#Find Team with rank 2 and 3  
print("Teams in whose Rank is 2 or 3 = ")  
print(df[df["Rank"].isin([2, 3])])  
#Find Teams 2014 or 2015  
print("Teams in year 2014 and 2015 = ")  
print(df[ ( df["Year"] == 2014) | ( df["Year"] == 2015 )])  
#Grouping on year and calculate mean of points  
grouped = df.groupby('Year')  
print(grouped['Points'].mean())  
#Grouping on Team and calculate mean of points  
grouped = df.groupby('Team')  
print(grouped['Points'].mean())  
#Maximum points in each year  
grouped = df.groupby('Year')  
print(grouped['Points'].max())

OUTPUT:

DataFrame =

Team Rank Year Points

0 Riders 1 2014 876

1 Riders 2 2015 789

2 Devils 2 2014 863

3 Devils 3 2015 673

4 Kings 3 2014 741

5 kings 4 2015 812

6 Kings 1 2016 756

7 Kings 1 2017 788

8 Riders 2 2016 694

9 Royals 4 2014 701

10 Royals 1 2015 804

11 Riders 2 2017 690

Teams in year 2014 =

Team Rank Year Points

0 Riders 1 2014 876

2 Devils 2 2014 863

4 Kings 3 2014 741

9 Royals 4 2014 701

Teams in whose Rank is 1 =

Team Rank Year Points

0 Riders 1 2014 876

6 Kings 1 2016 756

7 Kings 1 2017 788

10 Royals 1 2015 804

Teams in whose Rank is 2 or 3 =

Team Rank Year Points

1 Riders 2 2015 789

2 Devils 2 2014 863

3 Devils 3 2015 673

4 Kings 3 2014 741

8 Riders 2 2016 694

11 Riders 2 2017 690

Teams in year 2014 and 2015 =

Team Rank Year Points

0 Riders 1 2014 876

1 Riders 2 2015 789

2 Devils 2 2014 863

3 Devils 3 2015 673

4 Kings 3 2014 741

5 kings 4 2015 812

9 Royals 4 2014 701

10 Royals 1 2015 804

Year

2014 795.25

2015 769.50

2016 725.00

2017 739.00

Name: Points, dtype: float64

Team

Devils 768.000000

Kings 761.666667

Riders 762.250000

Royals 752.500000

kings 812.000000

Name: Points, dtype: float64

Year

2014 876

2015 812

2016 756

2017 788

Name: Points, dtype: int64

Process finished with exit code 0

1. **Create Data Frame for following data and apply following operations on data frame**

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **physics** | **chemistry** | **algebra** |
| **Somu** | **68** | **84** | **78** |
| **Kiku** | **74** | **56** | **88** |
| **Amol** | **77** | **73** | **82** |
| **Lini** | **78** | **69** | **87** |

* **Add new data for Geo whose marks is 87,92,97**
* **Add Maths marks for all students**
* **Sort data frame by name in ascending also descending**
* **Apply filter on name and physics**
* **Apply filter where column name start with chem**

import pandas as pd  
data = {'name': ['Somu', 'Kiku', 'Amol', 'Lini'],  
 'physics': [68, 74, 77, 78],  
 'chemistry': [84, 56, 73, 69],  
 'algebra': [78, 88, 82, 87]}  
#create dataframe  
df\_marks = pd.DataFrame(data)  
print('Original DataFrame\n------------------')  
print(df\_marks)  
#ADD row to the dataframe  
df\_marks.loc[len(df\_marks)]=['Geo', 87, 92,97]  
print("After adding new row = ")  
print(df\_marks)  
#Adding new column  
df\_marks.insert(2,"Maths",[45,66,78,90,91])  
print("After adding new column = ")  
print(df\_marks)  
#Sorting by name in ascending order  
df\_marks=df\_marks.sort\_values(by=['name'])  
print("Sorting Name by Ascending order = ",)  
print(df\_marks)  
#Sorting by name in decending order  
df\_marks=df\_marks.sort\_values(by=['name'], ascending=False)  
print("Sorting Name by Descending order = ",)  
print(df\_marks)  
#Filter on column name and physics  
df\_Phy=df\_marks.filter(items=['name','physics'])  
print("After Filter on column name and physics" )  
print(df\_Phy)  
#Filter Column start with chem  
df\_chem=df\_marks.filter(regex='^chem',axis=1)  
print("After Filter on chem" )  
print(df\_chem)

OUTPUT:

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ADMIN\PycharmProjects\pythonProject\exon.py

Original DataFrame

------------------

name physics chemistry algebra

0 Somu 68 84 78

1 Kiku 74 56 88

2 Amol 77 73 82

3 Lini 78 69 87

After adding new row =

name physics chemistry algebra

0 Somu 68 84 78

1 Kiku 74 56 88

2 Amol 77 73 82

3 Lini 78 69 87

4 Geo 87 92 97

After adding new column =

name physics Maths chemistry algebra

0 Somu 68 45 84 78

1 Kiku 74 66 56 88

2 Amol 77 78 73 82

3 Lini 78 90 69 87

4 Geo 87 91 92 97

Sorting Name by Ascending order =

name physics Maths chemistry algebra

2 Amol 77 78 73 82

4 Geo 87 91 92 97

1 Kiku 74 66 56 88

3 Lini 78 90 69 87

0 Somu 68 45 84 78

Sorting Name by Descending order =

name physics Maths chemistry algebra

0 Somu 68 45 84 78

3 Lini 78 90 69 87

1 Kiku 74 66 56 88

4 Geo 87 91 92 97

2 Amol 77 78 73 82

After Filter on column name and physics

name physics

0 Somu 68

3 Lini 78

1 Kiku 74

4 Geo 87

2 Amol 77

After Filter on chem

chemistry

0 84

3 69

1 56

4 92

2 73

Process finished with exit code 0

1. Demonstrate program for pandas string functions.

import pandas as pd  
import numpy as np  
s = pd.Series(['Tom', 'William Rick', 'John', 'Alber@t', np.nan, '1234','SteveSmith'])  
print("Series=")  
print(s)  
print("Series in lowercase=")  
print(s.str.lower())  
print("Series in uppercase=")  
print(s.str.upper())  
s = pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])  
print("new series =")  
print (s)  
print ("After Stripping:")  
print (s.str.strip())  
print(s.str.cat(sep='\_'))  
time\_sentences = ["Monday: The doctor's appointment is at 2:45 pm.",  
 "Tuesday: The dentist's appointment is at 11:30 am.",  
 "Wednesday: At 7:00 pm, there is a basketball game!",  
 "Thursday: Be back home by 11:15 pm at the latest.",  
 "Friday: Take the train at 08:10 am, arrive at 09:00am."]  
  
df = pd.DataFrame(time\_sentences, columns=['text'])  
print(df)  
# find which entries contain the word 'appointment'  
print("find which entries contain the word 'appointment")  
print(df[df['text'].str.contains('appointment')])  
# extract the entire time, the hours, the minutes, and the period  
print("extract the entire time, the hours, the minutes, and the period")  
print(df['text'].str.extractall(r'(?P<time>\d:\d{1,2})'))

OUTPUT:

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ADMIN\PycharmProjects\pythonProject\exonstrfn.py

Series=

0 Tom

1 William Rick

2 John

3 Alber@t

4 NaN

5 1234

6 SteveSmith

dtype: object

Series in lowercase=

0 tom

1 william rick

2 john

3 alber@t

4 NaN

5 1234

6 stevesmith

dtype: object

Series in uppercase=

0 TOM

1 WILLIAM RICK

2 JOHN

3 ALBER@T

4 NaN

5 1234

6 STEVESMITH

dtype: object

new series =

0 Tom

1 William Rick

2 John

3 Alber@t

dtype: object

After Stripping:

0 Tom

1 William Rick

2 John

3 Alber@t

dtype: object

Tom \_ William Rick\_John\_Alber@t

text

0 Monday: The doctor's appointment is at 2:45 pm.

1 Tuesday: The dentist's appointment is at 11:30...

2 Wednesday: At 7:00 pm, there is a basketball g...

3 Thursday: Be back home by 11:15 pm at the latest.

4 Friday: Take the train at 08:10 am, arrive at ...

find which entries contain the word 'appointment

text

0 Monday: The doctor's appointment is at 2:45 pm.

1 Tuesday: The dentist's appointment is at 11:30...

extract the entire time, the hours, the minutes, and the period

time

match

0 0 2:45

1 0 1:30

2 0 7:00

3 0 1:15

4 0 8:10

1 9:00

Process finished with exit code 0

1. Demonstrate merge function with left, right, outer in pandas.

import pandas as pd  
table1 = pd.DataFrame({"P\_ID" : (1,2,3,4,5,6,7,8),  
 "gender" : ("male", "male", "female","female",  
 "female", "male", "female", "male"),  
 "height" : (71,73,64,64,66,69,62,72),  
 "weight" : (175,225,130,125,165,160,115,250)})  
  
print(table1)  
table2 = pd.DataFrame({"P\_ID" : (1, 2, 4, 5, 7, 8, 9, 10),  
 "sex" : ("male", "male", "female","female",  
 "female", "male", "male", "female"),  
 "visits" : (1,2,4,12,2,2,1,1),  
 "checkup" : (1,1,1,1,1,1,0,0),  
 "follow\_up" : (0,0,1,2,0,0,0,0)  
 })  
print(table2)  
combined1 = pd.merge(table1, # First table  
 table2, # Second table  
 how="inner", # Merge method  
 on="P\_ID") # Column(s) to join on  
  
print(combined1)  
left\_join = pd.merge(table1, # First table  
 table2, # Second table  
 how="left", # Merge method  
 on="P\_ID") # Column(s) to join on  
  
print(left\_join)  
right\_join = pd.merge(table1, # First table  
 table2, # Second table  
 how="right", # Merge method  
 on="P\_ID") # Column(s) to join on  
  
print(right\_join)  
# An outer join keeps all key values in both data frames  
  
outer\_join = pd.merge(table1, # First table  
 table2, # Second table  
 how="outer", # Merge method  
 on="P\_ID") # Column(s) to join on  
  
print(outer\_join)

OUTPUT:

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ADMIN\PycharmProjects\pythonProject\25.py

P\_ID gender height weight

0 1 male 71 175

1 2 male 73 225

2 3 female 64 130

3 4 female 64 125

4 5 female 66 165

5 6 male 69 160

6 7 female 62 115

7 8 male 72 250

P\_ID sex visits checkup follow\_up

0 1 male 1 1 0

1 2 male 2 1 0

2 4 female 4 1 1

3 5 female 12 1 2

4 7 female 2 1 0

5 8 male 2 1 0

6 9 male 1 0 0

7 10 female 1 0 0

P\_ID gender height weight sex visits checkup follow\_up

0 1 male 71 175 male 1 1 0

1 2 male 73 225 male 2 1 0

2 4 female 64 125 female 4 1 1

3 5 female 66 165 female 12 1 2

4 7 female 62 115 female 2 1 0

5 8 male 72 250 male 2 1 0

P\_ID gender height weight sex visits checkup follow\_up

0 1 male 71 175 male 1.0 1.0 0.0

1 2 male 73 225 male 2.0 1.0 0.0

2 3 female 64 130 NaN NaN NaN NaN

3 4 female 64 125 female 4.0 1.0 1.0

4 5 female 66 165 female 12.0 1.0 2.0

5 6 male 69 160 NaN NaN NaN NaN

6 7 female 62 115 female 2.0 1.0 0.0

7 8 male 72 250 male 2.0 1.0 0.0

P\_ID gender height weight sex visits checkup follow\_up

0 1 male 71.0 175.0 male 1 1 0

1 2 male 73.0 225.0 male 2 1 0

2 4 female 64.0 125.0 female 4 1 1

3 5 female 66.0 165.0 female 12 1 2

4 7 female 62.0 115.0 female 2 1 0

5 8 male 72.0 250.0 male 2 1 0

6 9 NaN NaN NaN male 1 0 0

7 10 NaN NaN NaN female 1 0 0

P\_ID gender height weight sex visits checkup follow\_up

0 1 male 71.0 175.0 male 1.0 1.0 0.0

1 2 male 73.0 225.0 male 2.0 1.0 0.0

2 3 female 64.0 130.0 NaN NaN NaN NaN

3 4 female 64.0 125.0 female 4.0 1.0 1.0

4 5 female 66.0 165.0 female 12.0 1.0 2.0

5 6 male 69.0 160.0 NaN NaN NaN NaN

6 7 female 62.0 115.0 female 2.0 1.0 0.0

7 8 male 72.0 250.0 male 2.0 1.0 0.0

8 9 NaN NaN NaN male 1.0 0.0 0.0

9 10 NaN NaN NaN female 1.0 0.0 0.0

Process finished with exit code 0