**1. Given below is a dictionary having two keys ‘Boys’ and ‘Girls’ and having two lists of heights of five Boys and**

**Five Girls respectively as values associated with these keys**

**Original dictionary of lists:**

**{'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}**

**From the given dictionary of lists create the following list of dictionaries:**

**[{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69}, {'Boys': 69, 'Girls': 62}, {‘Boys’:74,**

**‘Girls’:61]**

**2. Write programs in Python using NumPy library to do the following:**

**a. Compute the mean, standard deviation, and variance of a two dimensional random integer array**

**along the second axis.**

**b. Get the indices of the sorted elements of a given array.**

**a. B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]**

**c. Create a 2-dimensional array of size m x n integer elements, also print the shape, type and data**

**type of the array and then reshape it into nx m array, n and m are user inputs given at the run time.**

**d. Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of**

**these elements in three separate arrays.**

**3. Create a dataframe having at least 3 columns and 50 rows to store numeric data generated using a random**

**function. Replace 10% of the values by null values whose index positions are generated using random function.**

**Do the following:**

**a. Identify and count missing values in a dataframe.**

**b. Drop the column having more than 5 null values.**

**c. Identify the row label having maximum of the sum of all values in a row and drop that row.**

**d. Sort the dataframe on the basis of the first column.**

**e. Remove all duplicates from the first column.**

**f. Find the correlation between first and second column and covariance between second and third**

**column.**

**g. Detect the outliers and remove the rows having outliers.**

**h. Discretize second column and create 5 bins**

**4. Consider two excel files having attendance of a workshop’s participants for two days. Each file has three**

**fields ‘Name’, ‘Time of joining’, duration (in minutes) where names are unique within a file. Note that duration**

**may take one of three values (30, 40, 50) only. Import the data into two dataframes and do the following:**

**a. Perform merging of the two dataframes to find the names of students who had attended the**

**workshop on both days.**

**b. Find names of all students who have attended workshop on either of the days.**

**c. Merge two data frames row-wise and find the total number of records in the data frame.**

**d. Merge two data frames and use two columns names and duration as multi-row indexes. Generate**

**descriptive statistics for this multi-index.**

**5. Taking Iris data, plot the following with proper legend and axis labels: (Download IRIS data from:**

**https://archive.ics.uci.edu/ml/datasets/iris or import it from sklearn.datasets)**

**a. Plot bar chart to show the frequency of each class label in the data.**

**b. Draw a scatter plot for Petal width vs sepal width.**

**c. Plot density distribution for feature petal length.**

**d. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.**

**6. Consider any sales training/ weather forecasting dataset**

**a. Compute mean of a series grouped by another series**

**b. Fill an intermittent time series to replace all missing dates with values of previous non-missing date.**

**c. Perform appropriate year-month string to dates conversion.**

**d. Split a dataset to group by two columns and then sort the aggregated results within the groups.**

**e. Split a given dataframe into groups with bin counts.**

**7. Consider a data frame containing data about students i.e. name, gender and passing division:**

**Name Birth\_Month Gender Pass\_Division**

**0 Mudit Chauhan December M III**

**1 Seema Chopra January F II**

**2 Rani Gupta March F I**

**3 Aditya Narayan October M I**

**4 Sanjeev Sahni February M II**

**5 Prakash Kumar December M III**

**6 Ritu Agarwal September F I**

**7 Akshay Goel August M I**

**8 Meeta Kulkarni July F II**

**9 Preeti Ahuja November F II**

**10 Sunil Das Gupta April M III**

**11 Sonali Sapre January F I**

**12 Rashmi Talwar June F III**

**13 Ashish Dubey May M II**

**14 Kiran Sharma February F II**

**15 Sameer Bansal October M I**

**a. Perform one hot encoding of the last two columns of categorical data using the get\_dummies() function.**

**b. Sort this data frame on the “Birth Month” column (i.e. January to December). Hint: Convert Month to**

**Categorical.**

**8. Consider the following data frame containing a family name, gender of the family member and her/his monthly**

**income in each record.**

**Name Gender MonthlyIncome (Rs.)**

**Shah Male 114000.00**

**Vats Male 65000.00**

**Vats Female 43150.00**

**Kumar Female 69500.00**

**Vats Female 155000.00**

**Kumar Male 103000.00**

**Shah Male 55000.00**

**Shah Female 112400.00**

**Kumar Female 81030.00**

**Vats Male 71900.00**

**Write a program in Python using Pandas to perform the following:**

**a. Calculate and display familywise gross monthly income.**

**b. Calculate and display the member with the highest monthly income in a family.**

**c. Calculate and display monthly income of all members with income greater than Rs. 60000.00.**

**d. Calculate and display the average monthly income of the female members in the Shah family**

Q1 Given below is a dictionary having two keys ‘Boys’ and ‘Girls’ and having two lists of heights of five Boys and Five Girls respectively as values associated with these keys

Original dictionary of lists:

{'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}

From the given dictionary of lists create the following list of dictionaries:

[{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69}, {'Boys': 69, 'Girls': 62}, {‘Boys’:74, ‘Girls’:61]

def list\_of\_dict(heights):

keys=heights.keys()

# print(keys)

values = zip(\*[heights[k] for k in keys])

# print(values)

result = [dict(zip(keys,v )) for v in values]

return result

heights = {'Boys':[72,68,70,69,74], 'Girls':[63,65,69,62,61]}

print("\n ORIGINAL DICTIONARY OF LISTS :" , heights)

print("\n NOW LIST OF DICTIONARIES : \n",list\_of\_dict(heights))

**Q2 Write programs in Python using NumPy library to do the following:**

**a. Compute the mean, standard deviation, and variance of a two dimensional random integer array along the second axis.**

**b. Get the indices of the sorted elements of a given array.**

**a. B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]**

**c. Create a 2-dimensional array of size m x n integer elements, also print the shape, type and data type of the array and then reshape it into nx m array, n and m are user inputs given at the run time.**

**d. Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.**

In [2]:

part a

#part a

arr = np.random.randint(1,50,(4,6))

arr

#along the second axis

#Mean

print('Mean of the array: ',arr.mean(axis=1))

#standard deviation

print('Standard Deviation of the array: ',arr.std(axis=1))

#variance

print('Variance of the array: ',arr.var(axis=1))

#part b

B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]

arr1 = np.array(B)

#arr1

print("Sorted array: ",np.sort(arr1))

print("Indices of the sorted elements of a given array: ",np.argsort(arr1))

#part c

m = int(input('Enter the number of rows(m): '))

n = int(input('Enter the number of columns(n): '))

arr2 = np.random.randint(1,100,(m,n))

print(arr2)

print('Shape: ',arr2.shape)

print('Type: ',type(arr2))

print('Data Type: ',arr2.dtype)

arr2 = arr2.reshape(n,m)

print('After reshaping: \n',arr2)

print('New Shape: ',arr2.shape)

#part D

x = np.array([1, 0, 3, 4])

print("ORIGINAL ARRAY ::-> ",x)

print("\nTest if none of the elements of the said array is zero ::-> ", np.all(x))

res = np.where(x == 0)[0]

print("The index of the zero elements is :: ",res)

x = np.array([1, 0, 0, 3, 2, 0])

print("\n")

print("\nORIGINAL ARRAY ::-> ",x)

print("\nTest whether any of the elements of a given array is non-zero ::",np.any(x))

res = np.where(x != 0)[0]

print("The index of the non- zero elements is :: ",res)

x = np.array([0, 0, 0, 0])

a = np.array([1, 0, np.nan, 3, np.nan])

print("\n")

print("\nORIGINAL ARRAY ::-> ",a)

print("\nTest element-wise for NaN :: ",np.isnan(a))

res = np.where(np.isnan(a) == True)[0]

print("The index of the zero elements is :: ",res)

**3 Create a dataframe having at least 3 columns and 50 rows to store numeric data generated using a random function. Replace 10% of the values by null values whose index positions are generated using random function. Do the following:**

* **a. Identify and count missing values in a dataframe.**
* **b. Drop the column having more than 5 null values.**
* **c. Identify the row label having maximum of the sum of all values in a row and drop that row.**
* **d. Sort the dataframe on the basis of the first column.**
* **e. Remove all duplicates from the first column.**
* **f. Find the correlation between first and second column and covariance between second and third column.**
* **g. Detect the outliers and remove the rows having outliers.**

**h. Discretize second column and create 5 bins**  
  
**import** pandas **as** pd

**import** numpy **as** np

df **=** pd**.**DataFrame(np**.**random**.**randint(0,100,size**=**(50,3)), columns**=**list('123'))

df**.**head()

**for** c **in** df**.**sample(int(df**.**shape[0]**\***df**.**shape[1]**\***0.10))**.**index:

df**.**loc[c,str(np**.**random**.**randint(1,4))]**=**np**.**nan

df

*#part A*

print(df**.**isnull()**.**sum()**.**sum())

15

#PART B

**for** col **in** df**.**columns:

print(col,df[col]**.**isnull()**.**sum())

df**.**dropna(axis **=** 1,thresh**=**(df**.**shape[0]**-**5))**.**head()

#PART C

sum**=**df**.**sum(axis**=**1)

print("SUM IS :\n",sum)

print("\nMAXIMUM SUM IS :",sum**.**max())

max\_sum\_row **=** df**.**sum(axis**=**1)**.**idxmax()

print("\nRow index having maximum sum is :" ,max\_sum\_row)

df **=** df**.**drop(max\_sum\_row ,axis **=**0)

print("\nDATA Frame AFTER REMOVING THE ROW HAVING MAXIMUM SUM VALUE")

df

*#part D*

sortdf**=**df**.**sort\_values('1')

sortdf**.**head()

*# PART E*

df **=**df**.**drop\_duplicates(subset**=**'1',keep **=** "first")

print(df)

*#part F*

correlation **=** df['1']**.**corr(df['2'])

print("CORRELATION between column 1 and 2 : ", correlation)

covariance **=** df['2']**.**cov(df['3'])

print("COVARIANCE between column 2 and 3 :",covariance)

*#part G*

df**.**plot**.**box()

*#part H*

df1 **=** pd**.**cut(df['2'],bins**=**5)**.**head()

df1

Q4 Consider two excel files having attendance of a workshop’s participants for two days. Each file has three fields ‘Name’, ‘Time of joining’, duration (in minutes) where names are unique within a file. Note that duration may take one of three values (30, 40, 50) only. Import the data into two dataframes and do the following:

a.Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.

b. Find names of all students who have attended workshop on either of the days.

c. Merge two data frames row-wise and find the total number of records in the data frame.

d. Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index.

*#first we are importing the essential libraries*

**import** numpy **as** np

**import** pandas **as** pd

*#taking the data from excel files*

filename**=**'C:/Users/shadj/Desktop/Shad/DAV/Practicals/Book1.xlsx'

df1**=**pd**.**read\_excel(filename)

df1

df2**=**pd**.**read\_excel(filename,sheet\_name**=**'Sheet2')

df2

*#* ***a)****Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.*

pd**.**merge(dfDay1,dfDay2,how**=**'inner',on**=**'Name')

*#***b)***Find names of all students who have attended workshop on either of the days.*

either\_day **=** pd**.**merge(dfDay1,dfDay2,how**=**'outer',on**=**'Name')

either\_day

*#****c)****Merge two data frames row-wise and find the total number of records in the data frame. #using the either day from part b*

either\_day['Name']**.**count()

**PART D**

*# Merge two data frames and use two columns names and duration as multi-row indexes.*

*#Generate descriptive statistics for this multi-index*

both\_days **=** pd**.**merge(dfDay1,dfDay2,how**=**'outer',on**=**['Name','Duration'])**.**copy() *# creates a copy of an existing list*

both\_days**.**fillna(value**=**'-',inplace**=True**) *# to fill out the missing values in the given series object*

both\_days**.**set\_index(['Name','Duration']) *# a method to set a List as index of a Data Frame*

**Q5. Taking Iris data, plot the following with proper legend and axis labels: (Download IRIS data from: https://archive.ics.uci.edu/ml/datasets/iris or import it from sklearn.datasets)**

**a. Plot bar chart to show the frequency of each class label in the data.**

**b. Draw a scatter plot for Petal width vs sepal width.**

**c. Plot density distribution for feature petal length.**

**d. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.**

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

iris **=** sns**.**load\_dataset('iris')

*#part A*

sns**.**countplot(x**=**'species',data**=**iris,palette**=**'Set2')

plt**.**xlabel('Species')

plt**.**ylabel('Frequency')

plt**.**title('Frequency of Each class label')

*#part B*

plt**.**scatter(x**=**'petal\_width',y**=**'sepal\_width',data**=**iris)

plt**.**xlabel('Petal Width')

plt**.**ylabel('Sepal Width')

plt**.**title("Scatter plot Petel width vs Sepal Width")

*#part C*

sns**.**histplot(iris['petal\_length'],kde**=False**,bins**=**30)

*#part D*

sns**.**pairplot(iris,hue**=**'species',palette**=**'coolwarm')

**6. Consider any sales training/ weather forecasting dataset**

**a. Compute mean of a series grouped by another series**

**b. Fill an intermittent time series to replace all missing dates with values   
 of previous non-missing date.**

**c. Perform appropriate year-month string to dates conversion.**

**d. Split a dataset to group by two columns and then sort the aggregated**

**results within the groups.**

**e. Split a given dataframe into groups with bin counts.**

**import** numpy **as** np

**import** pandas **as** pd

data**=**pd**.**read\_csv("weatherHistory.csv")

fram**=**data**.**head(15)

frame**=**fram**.**drop(["Formatted Date","Summary",'Loud Cover','Pressure (millibars)', 'Daily Summary'],axis**=**1)

x**=**pd**.**date\_range('01/08/23',periods**=**15);

frame**.**insert(value**=**x,column**=**"Date",loc**=**0)

frame

a) Compute the mean of a series grouped by another series :

In [2]:

frame**.**groupby("Humidity")["Visibility (km)"]**.**mean(numeric\_only**=True**)

b)Fill an intermittent time series to replace all missing dates with values of previous non-missing date.

value\_to\_replace**=**pd**.**date\_range("01/10/2023",periods**=**5)

frame**.**replace(value\_to\_replace,np**.**nan,inplace**=True**)

frame

frame["Date"]**.**replace(np**.**nan,method**=**'ffill',inplace**=True**)

frame

c)Perform appropriate year-month string to dates conversion.

**import** datetime **as** dt

frame["Date"]**=**pd**.**to\_datetime(frame["Date"])**.**dt**.**strftime("%d-%m-%Y")

frame

d)Split a dataset to group by two columns and then sort the aggregated results within the groups.

grp**=**frame**.**groupby(["Humidity","Visibility (km)"])

grp**.**agg({"Wind Speed (km/h)":'mean'})**.**sort\_values(by**=**"Wind Speed (km/h)")

e)Split a given dataframe into groups with bin counts.

grp2**=**frame**.**groupby(['Visibility (km)', pd**.**cut(frame['Temperature (C)'],4)])

grp2**.**size()**.**unstack()

**7. Consider a data frame containing data about students i.e. name, gender and passing division:**

**Name Birth\_Month Gender Pass\_Division**

**0 Mudit Chauhan December M III**

**1 Seema Chopra January F II**

**2 Rani Gupta March F I**

**3 Aditya Narayan October M I**

**4 Sanjeev Sahni February M II**

**5 Prakash Kumar December M III**

**6 Ritu Agarwal September F I**

**7 Akshay Goel August M I**

**8 Meeta Kulkarni July F II**

**9 Preeti Ahuja November F II**

**10 Sunil Das Gupta April M III**

**11 Sonali Sapre January F I**

**12 Rashmi Talwar June F III**

**13 Ashish Dubey May M II**

**14 Kiran Sharma February F II**

**15 Sameer Bansal October M I**

**a. Perform one hot encoding of the last two columns of categorical data using the get\_dummies() function.**

**b. Sort this data frame on the “Birth Month” column (i.e. January to December). Hint: Convert Month to**

**Categorical.**

**import** numpy **as** np

**import** pandas **as** pd

frame**=** pd**.**read\_table('data\_7.txt',sep**=**",")

frame

a. Perform one hot encoding of the last two columns of categorical data using the get\_dummies() function.

ohe1**=**pd**.**get\_dummies(frame["Gender"])

ohe2**=**pd**.**get\_dummies(frame["Pass\_Division"])

print(ohe1, ohe2, sep**=**"\n\n\n")

b)Sort this data frame on the “Birth Month” column (i.e. January to December). Hint: Convert Month to Categorical.

frame["Birth\_Month"]**=**pd**.**Categorical(frame["Birth\_Month"],categories**=**["January","February","March","April","May","June","July","August","September","October","November","December"])

frame**.**sort\_values(by**=**"Birth\_Month")

**8. Consider the following data frame containing a family name, gender of the family member and her/his monthly**

**income in each record.**

**Name Gender MonthlyIncome (Rs.)**

**Shah Male 114000.00**

**Vats Male 65000.00**

**Vats Female 43150.00**

**Kumar Female 69500.00**

**Vats Female 155000.00**

**Kumar Male 103000.00**

**Shah Male 55000.00**

**Shah Female 112400.00**

**Kumar Female 81030.00**

**Vats Male 71900.00**

**Write a program in Python using Pandas to perform the following:**

**a. Calculate and display familywise gross monthly income.**

**b. Calculate and display the member with the highest monthly income in a family.**

**c. Calculate and display monthly income of all members with income greater than Rs. 60000.00.**

**d. Calculate and display the average monthly income of the female members in the Shah family**

**import** numpy **as** np

**import** pandas **as** pd

data**=**{

"Name":["Shah","Vats","Vats","Kumar","Vats","Kumar","Shah","Shah","Kumar","Vats"],

"Gender":["Male","Male","Female","Female","Female","Male","Male","Female","Female","Male"],

"Salary":[114000.00,65000.00,43150.00,69500.00,155000.00,103000.00,55000.00,112400.00,81030.00,71900.00]

}

frame**=**pd**.**DataFrame(data)

frame

a)Calculate and display family wise gross monthly income

In [2]:

grp**=**frame**.**groupby("Name")

grp**.**apply(sum)**.**Salary

b)Calculate and display the member with the highest monthly income in a family

In [4]:

grp**.**max()

c)Calculate and display monthly income of all members with income greater than Rs. 60000.00.

In [5]:

frame[frame**.**Salary**>**60000]

d)Calculate and display the average monthly income of the female members in the Shah family.

In [7]:

f**=**frame**.**loc[frame["Name"]**==**"Shah"]

f2**=**f**.**loc[f["Gender"]**==**"Female"]

f2**.**mean(numeric\_only**=True**)