DAV practical file

Ques-1

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

FiveGirls 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]

Code:-

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

result_list=[{'Boys':b,'Girls':g} for b,g in zip(original_dict['Boys'],original_dict['Girls'])]

result_list
```

Ques-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.

Code:-a

```
import numpy as np

# Create a random 2D array with shape (m, n)

m, n = 3, 4 # You can change m and n as needed

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

print(random_array)
```

```
# Calculate mean, standard deviation, and variance along the second axis

(axis=1)

mean = np.mean(random_array, axis=1)

std_dev = np.std(random_array, axis=1)

variance = np.var(random_array, axis=1)

print("Mean along the second axis:", mean)

print("Standard Deviation along the second axis:", std_dev)

print("Variance along the second axis:", variance)
```

Code:-b

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

# Get the indices that would sort the array

sorted_indices = np.argsort(B)

print("Indices of sorted elements:", sorted_indices)
```

Code:-c

```
import numpy as np

# Get user inputs for n and m

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

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

# Create a random 2D array of size m x n

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

# Print shape, type, and data type of the array

print("Shape of the array:", random_array.shape)

print("Type of the array:", type(random_array))

print("Data type of the elements:", random_array.dtype)

# Reshape the array into nxm

reshaped_array = random_array.reshape(n, m)

print("Reshaped array:")

print(reshaped_array)
```

Code:-D

```
import numpy as np

# Create an example array

array = np.array([0, 1, 2, 0, np.nan, 4, 5, np.nan])

# Test for zero elements
```

```
zero_indices = np.where(array == 0)

# Test for non-zero elements

nonzero_indices = np.where(array != 0)

# Test for NaN elements

nan_indices = np.isnan(array)

print("Indices of zero elements:", zero_indices)

print("Indices of non-zero elements:", nonzero_indices)

print("Indices of NaN elements:", np.where(nan_indices))
```

Ques-3

Question-Create a dataframe having at least 3 columns and 50 rows to store numericdata 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 2 nd & 3 rd column.

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

h. Discretize second column and create 5 b.

Code: (a)

```
import numpy as np

import pandas as pd

import math


firstdata=np.random.randint(1,100,size=(50,3))

df=pd.DataFrame(firstdata,columns=['x','Y','Z'])

size2=len(df)*0.1

size2=int(size2)
```

```
for i in range(0,(size2*3)):

df.iat[np.random.randint(1,50,),1]=float("nan") count_nan=0

for i in range(0,len(df.columns)):

for j in range(0,len(df)):

if(math.isnan(df.iat[j,i])):

count_nan=count_nan+1

print("Number of MissingValues :- ",count_nan)
```

Code(b)

```
for i in range(0,len(df.columns)):

count_num=0

for j in range(0,len(df)):

if(math.isnan(df.iat[j,i])):

count_num=count_num+1


if(count_num>5):

df.drop(df.columns[[i]],axis=1,inplace=True)

print("Dropped Column ",i)
```

Code(c)

```
arr=[] for j in range(0,len(df)):

count_num=0

for i in range(0,len(df.columns)):

count_num=count_num+df.iat[j,i]

arr.append(count_num)

x=np.argsort(arr)

x=x[len(x)-1]

df.drop(x)

print("Dropped row with sum of elements as")
```

Code(D)

```
df.sort_values(by=df.columns[0])
```

Code(e)

```
for i in range(0,len(df.columns)):

count_num=0

for j in range(0,len(df)):
```

```
if(math.isnan(df.iat[j,i])):

count_num=count_num+1

if(count_num>5):

df.drop(df.columns[[i]],axis=1,inplace=True)

print("Dropped Column ",i)
```

Code(F)

```
print("Correlation between 1st and 2nd column is :- ", np.corrcoef(df.index, df["x"]));

print("Covariance between2nd and 3rd column is :- ", np.corrcoef(df["x"], df["Z"]));
```

Code(g)

```
 import seaborn as sns

df['x'][22]=200

sns.boxplot(x=df['x'])

ol=np.where(df['x']>100)

print(ol)

df.drop(ol[0])
```

Code(h)

```
edges=[0,20,40,60,80,100]

temp=pd.cut(df.iloc[:,2],edges)

print(temp)
```

Ques-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.

Code:-

```
import numpy as np
import pandas as pd
df1=pd.read_excel("first.xlsx")
df2=pd.read_excel("second.xlsx")
print(df2)
```

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

```
either1=pd.merge(df1,df2,how='outer',on='Name')
```

either1

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

either1=pd.merge(df1,df2,how='outer',on='Name')

either1

either1['Name'].count()

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

descriptive statistics for this multi-index.

both_day=pd.merge(df1,df2,how="outer",on=['Name',"Duration"]).copy()

both_day

both_day.set_index(['Name',"Duration"])

Question-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.

import pandas as pd

import numpy as np

from sklearn import datasets

import matplotlib.pyplot as plt

data2=pd.read_csv('C:\\ProgramData\\Anaconda3\\lib\\sitepackages\\

sklearn\\datasets\\data\\iris.csv')

data1 = datasets.load_iris()

data2

data2.plot.bar(color=["red",'blue',"green",'gray'])

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

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import preprocessing

import seaborn as sns

df=pd.read_csv("iris.csv")

df

sns.regplot(x=df['petal.width'],y=df['sepal.width'],color='gray')

c. Plot density distribution for feature petal length.

d. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.
sns.pairplot(df)

sns.pairplot(df,diag_kind='kde',plot_kws={'alpha':0.2})

Q6. Consider any sales training/ weather forecasting dataset

```
import numpy as np

import pandas as pd

df=pd.read_csv('weather1.csv')

df
```

a. Compute mean of a series grouped by another series

```
df1=df.groupby('outlook')

df1

df1.mean()
```

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

```
df.loc[df['temperature']==31,'temperature']=np.nan
df

df=df.fillna(method='ffill')

df
```

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

```
import datetime
dtr=df['Date']
df['date_formate']=pd.to_datetime(df['Date']).dt.to_period('m')
df
```

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

```
sdf=df.groupby(['humidity','date_formate']).agg({'temperature':'mean'}).sort_

values(['humidity','date_formate'])

sdf
```

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

```
bins=pd.cut(df['temperature'],bins=[0,29,37,41],labels=['Low','Medium','High'
])
gr=df.groupby(bins)
gr
for humidity,humidity1 in gr:
print(f" BIN:{humidity}")
print(humidity1)
```

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 pandas as pd data = {

'Name': ['Mudit Chauhan', 'Seema Chopra', 'Rani Gupta', 'Aditya Narayan', 'Sanjeev Sahni', 'Prakash Kumar', 'Ritu Agarwal', 'Akshay Goel', 'Meeta Kulkarni', 'Preeti Ahuja',

'Sunil Das Gupta', 'Sonali Sapre', 'Rashmi Talwar', 'Ashish Dubey', 'Kiran Sharma', 'Sameer 'Birth_Month': ['December', 'January', 'March', 'October', 'February', 'December', 'September 'August', 'July', 'November', 'April', 'January', 'June', 'May', 'February', 'October'],

'Gender': ['M', 'F', 'F', 'M', 'M', 'M', 'F', 'M', 'F', 'F', 'M', 'F', 'F', 'M', 'F', 'M'],

'Pass_Division': ['III', 'II', 'I', 'I', 'II', 'III', 'I', 'I', 'II', 'II', 'III', 'I', 'III', 'II', 'II', 'I']

}

df = pd.DataFrame(data) df

a.df_encoded **=** pd.get_dummies(df, columns**=**['Gender', 'Pass_Division']) df_encoded

b.month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'Septembe
'October', 'November', 'December']
df_encoded['Birth_Month'] = pd.Categorical(df_encoded['Birth_Month'], categories=month_o
ordered=True)
df_sorted = df_encoded.sort_values('Birth_Month')
print("DataFrame after one-hot encoding and sorting by Birth Month:\n", df_sorted)

Question8

Consider the following data frame containing a family name, gender of the family member and her/his
monthly income in each record.

a. Calculate and display familywise gross monthly income.

```
import pandas as pd
import numpy as np
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]
}
df=pd.DataFrame(data)
df
```

```
gross_salary=df['Salary'].groupby(df['Name'])

gross_salary.sum()
```

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

```
df1=gross_salary.max()
df1
```

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

```
df2=df[df['Salary']>60000]

df2[["Name",'Salary']]
```

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

```
df3=df[(df['Gender']=='Female')&(df['Name']=='vats')]

print(df3)

df3=df3.groupby(df['Name'])

df3.mean()
```

Project

```
import pandas as pd
df = pd.read_csv("covid_19_india.csv")
```

```
print(df)

import matplotlib.pyplot as plt

# Convert the 'Date' column to datetime format

df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')

# Task 1: For each state, find maximum cases reported for confirmed, deaths, and recovered # for any three months of the year 2020.

# Filter data for the year 2020

df_2020 = df[(df['Date'] >= '2020-01-01') & (df['Date'] < '2021-01-01')]

# Group by state and find maximum cases for confirmed, deaths, and recovered

result = df_2020.groupby(['State/UnionTerritory', 'Date']).agg({ 'Confirmed': 'max',

'Deaths': 'max',

'Cured': 'max'

}).reset_index()
```

```
# Display the result

print("Task 1: Maximum cases reported for confirmed, deaths, and recovered individually for print(result)


# Filter data for the specified states

selected_states = ['Karnataka', 'Gujarat', 'Haryana', 'Uttar Pradesh'] df_selected_states = df[df['State/UnionTerritory'].isin(selected_states)]

# Group by state and month, then sum the cured cases

df_cured_monthly = df_selected_states.groupby([df_selected_states['Date'].dt.to_period("M" 'Cured': 'sum'

}).unstack()

# Plot the subplots

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10), sharex=True)

for i, state in enumerate(selected_states): df_cured_monthly['Cured'][state].plot(ax=axes[i // 2, i % 2], marker='o') axes[i // 2, i % 2].set_title(state)
```

```
plt.suptitle('Total Number of Cured Cases Month-wise (April 2020 to March 2021)') plt.xlabel('Month')

plt.ylabel('Total Cured Cases') plt.show()


# Filter data for the specified states and months

selected_states_task3 = ['Karnataka', 'Delhi', 'Madhya Pradesh']

df_task3 = df[df['State/UnionTerritory'].isin(selected_states_task3) & (df['Date'].dt.month.isin(

# Group by state and sum the deaths

df_task3_grouped = df_task3.groupby(['State/UnionTerritory', 'Date']).agg({'Deaths': 'sum'}).r

# Pivot the table for plotting

df_task3_pivot = df_task3_grouped.pivot(index='Date', columns='State/UnionTerritory', value

# Plot the stacked bar chart
```

```
df_task3_pivot.plot(kind='bar', stacked=True)
plt.title('Comparison of Deaths in May 2020 and May 2021') plt.xlabel('Date')
plt.ylabel('Total Deaths') plt.show()


# Filter data for Uttar Pradesh
df_up = df[df['State/UnionTerritory'] == 'Uttar Pradesh']
# Group by month and calculate the correlation
correlation_df = df_up.groupby(df_up['Date'].dt.to_period("M")).agg({
'Confirmed': 'sum',
'Deaths': 'sum'
})
# Plot the graph
plt.figure(figsize=(10, 6))
plt.plot(correlation_df.index.astype(str), correlation_df['Confirmed'], label='Confirmed Cases',
plt.plot(correlation_df.index.astype(str), correlation_df['Deaths'], label='Deaths', marker='o')
```

```
plt.title('Month-wise Relation between Confirmed Cases and Deaths in Uttar Pradesh') plt.xlabel('Month')

plt.ylabel('Number of Cases') plt.legend()

plt.show()
```