VIT-AP UNIVERSITY, ANDHRA PRADESH

CSE4005 – DWDM - LAB 6

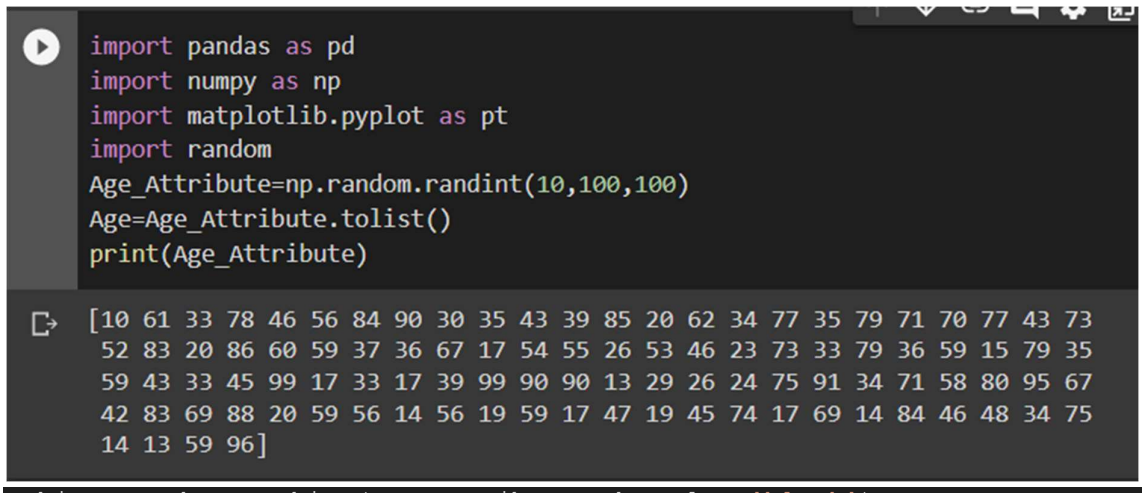
Academic year: 2022-2023 **Branch/ Class:** B.Tech
Semester: Fall **Date: 09/10/22**
Faculty Name: Dr. Aravapalli Rama Satish **School:** SCOPE
Student name: MAJJIGA JASWANTH **Reg. no.: 20BCD7171**

---

1) a.Generate 100 random numbers for the age attribute and plot the equal-width(uniform) and equal frequency histograms.
b.Add two attributes ID (1, 2, 3…, 100), Category (youth, middle_aged, Senior) for each age value. Develop user defined functions to perform sampling of the age attribute: SRSWOR, SRSWR, and stratified sampling. Use samples of size 10 and the strata "youth," "middle-aged," and "senior."

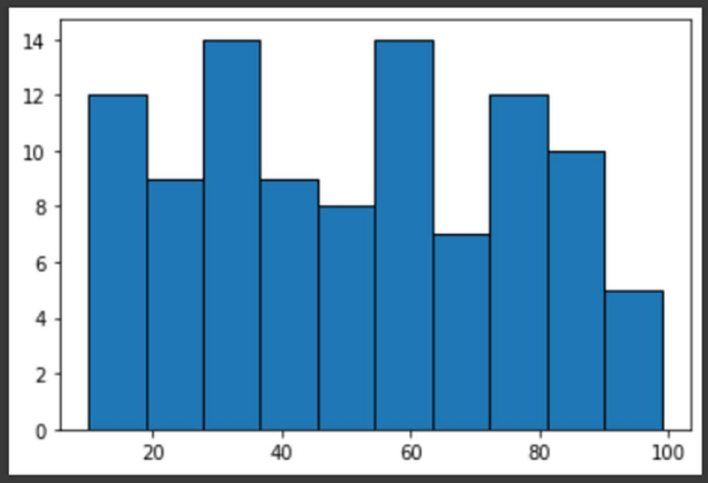
Code and output:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pt
import random
Age_Attribute=np.random.randint(10,100,100)
Age=Age_Attribute.tolist()
print(Age_Attribute)
```



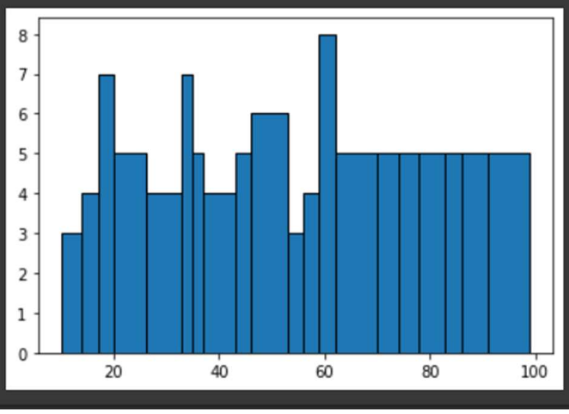

```
n,bins,patches=pt.hist(Age_Attribute,edgecolor='black')
pt.show()
```

```
n,bins,patches=pt.hist(Age_Attribute,edgecolor='black')
pt.show()
```




```
def equalfree(x,nbin):
  nlen = len(x)
  return np.interp(np.linspace(0,nlen,nbin +1),
                   np.arange(nlen),
                   np.sort(x))
n,bins,patches = pt.hist(Age_Attribute,equalfree(Age_Attribute,20),edge
color='black')
pt.show()
```

```
def equalfree(x,nbin):
  nlen = len(x)
  return np.interp(np.linspace(0,nlen,nbin +1),
                   np.arange(nlen),
                   np.sort(x))
n,bins,patches = pt.hist(Age_Attribute,equalfree(Age_Attribute,20),edgecolor='black')
pt.show()
```




```
def SRSWR(n):
  b=[random.choice(Age_Attribute) for i in range(n)]
  return b
SRSWR(20)
```

```
def SRSWR(n):
  b=[random.choice(Age_Attribute) for i in range(n)]
  return b
SRSWR(20)
```

```
[20,
 34,
 75,
 13,
 33,
 20,
 33,
 23,
 13,
 20,
 91,
 43,
 34,
 24,
 46,
 34,
 59,
 35,
 14,
 90]
```

```
Category=[]
for i in range(len(Age_Attribute)):
  if Age_Attribute[i] > 60:
    Category.append("Senior")
  elif Age_Attribute[i]>=35 and Age_Attribute[i]<=50:
    Category.append("Middle-aged")
  else:
    Category.append("Youth")
print(Category)
```

```
Category=[]
for i in range(len(Age_Attribute)):
  if Age_Attribute[i] > 60:
    Category.append("Senior")
  elif Age_Attribute[i]>=35 and Age_Attribute[i]<=50:
    Category.append("Middle-aged")
  else:
    Category.append("Youth")
print(Category)
```

```
['Youth', 'Senior', 'Youth', 'Senior', 'Middle-aged', 'Youth', 'Senior', 'Senior', 'Youth',
```

```
import pandas as pd
def Stratified():
  d=pd.DataFrame({'Age_Attribute':Age_Attribute,'Category':Category})
  A=d.groupby('Category',group_keys=False).apply(lambda x: x.sample(fra
c=0.40))
  print(A)
Stratified()
```

```
import pandas as pd
def Stratified():
  d=pd.DataFrame({'Age_Attribute':Age_Attribute,'Category':Category})
  A=d.groupby('Category',group_keys=False).apply(lambda x: x.sample(frac=0.40))
  print(A)
Stratified()
```

	Age_Attribute	Category
84	47	Middle-aged
56	39	Middle-aged
11	39	Middle-aged
4	46	Middle-aged
9	35	Middle-aged
47	35	Middle-aged
93	48	Middle-aged
22	43	Middle-aged
64	75	Senior
67	71	Senior
19	71	Senior
57	99	Senior
52	99	Senior
6	84	Senior
7	90	Senior
99	96	Senior
32	67	Senior
18	79	Senior
71	67	Senior
25	83	Senior
21	77	Senior
70	95	Senior
62	26	Youth
0	10	Youth
37	53	Youth
90	14	Youth
45	15	Youth
79	14	Youth
68	58	Youth
81	19	Youth
36	26	Youth
80	56	Youth
60	13	Youth
53	17	Youth
15	34	Youth
33	17	Youth
5	56	Youth

2) Develop user defined function to calculate chi-square correlation test for Nominal Data and to decide whether the two nominal attributes are independent or not.
Example Data: (Level of significance:0.05)

Apply your function all pairs of Nominal attributes and formulate a chi-square correlation matrix and label each cell as ‘D’(Dependent) or ‘I’(Independent).

	High School	Bachelors	Masters	Ph.d.	Total
Female	90	84	76	66	316
Male	60	64	73	83	280
Total	150	148	149	149	596

df	0.5	0.10	0.05	0.02	0.01	0.001
1	0.455	2.706	3.841	5.412	6.635	10.827
2	1.386	4.605	5.991	7.824	9.210	13.815
3	2.366	6.251	7.815	9.837	11.345	16.268
4	3.357	7.779	9.488	11.668	13.277	18.465
5	4.351	9.236	11.070	13.388	15.086	20.517

Code and output:

```
import pandas as pd
import numpy as np
def chi(observed,expected):
  chi=list()
  for i in observed:
    for k,j in observed.iterrows():
      if(i!='TOTAL' and k!='TOTAL'):
        print(i)
        print(k)
        print(observed[i][k])
        print(expected[i][k])
        a=observed[i][k]-expected[i][k]
        a=float(a*a/expected[i][k])
    if a<3.841:
      observed[i][k]='I'
    else:
      observed[i][k]='D'
  print(observed)
DATA={'HIGH SCHOOL':[90,60],'BACHELORS':[84,64],'MASTERS':[76,73],'PH.D':[6
6,83]}
DATA=pd.DataFrame(DATA,index=pd.Index(['Female','Male']))
DATA['TOTAL']=DATA.sum(axis=1)
DATA.loc['TOTAL']=DATA.sum(axis=0)
DATA2=DATA.copy()
for i in DATA2:
  for k,j in DATA.iterrows():
    if(i!='TOTAL' and k!='TOTAL'):
      DATA2[i][k]=DATA[i]['TOTAL']*DATA.loc[k]['TOTAL']/DATA['TOTAL']['TOTA
L']
print(DATA)
print(DATA2)
chi(DATA,DATA2)
```

```
import pandas as pd
import numpy as np
def chi(observed,expected):
  chi=list()
  for i in observed:
    for k,j in observed.iterrows():
      if(i!='TOTAL' and k!='TOTAL'):
        print(i)
        print(k)
        print(observed[i][k])
        print(expected[i][k])
        a=observed[i][k]-expected[i][k]
        a=float(a*a/expected[i][k])
    if a<3.841:
      observed[i][k]='I'
    else:
      observed[i][k]='D'
  print(observed)
DATA={'HIGH SCHOOL':[90,60],'BACHELORS':[84,64],'MASTERS':[76,73],'PH.D':[66,83]}
DATA=pd.DataFrame(DATA,index=pd.Index(['Female','Male']))
DATA['TOTAL']=DATA.sum(axis=1)
DATA.loc['TOTAL']=DATA.sum(axis=0)
DATA2=DATA.copy()
for i in DATA2:
  for k,j in DATA.iterrows():
    if(i!='TOTAL' and k!='TOTAL'):
      DATA2[i][k]=DATA[i]['TOTAL']*DATA.loc[k]['TOTAL']/DATA['TOTAL']['TOTAL']
print(DATA)
print(DATA2)
chi(DATA,DATA2)
```

```
        HIGH SCHOOL  BACHELORS  MASTERS  PH.D  TOTAL
Female           90         84       76    66    316
Male             60         64       73    83    280
TOTAL           150        148      149   149    596
        HIGH SCHOOL  BACHELORS  MASTERS  PH.D  TOTAL
Female           79         78       79    79    316
Male             70         69       70    70    280
TOTAL           150        148      149   149    596
HIGH SCHOOL
Female
90
79
HIGH SCHOOL
Male
60
70
BACHELORS
Female
84
78
BACHELORS
Male
64
69
MASTERS
Female
76
79
MASTERS
Male
73
70
PH.D
Female
66
79
PH.D
Male
83
70
      HIGH SCHOOL BACHELORS MASTERS PH.D TOTAL
Female          90        84      76   66   316
Male            60        64      73   83   280
TOTAL            I         I       I    I     I
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
  from ipykernel import kernelapp as app
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1732: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
  self._setitem_single_block(indexer, value, name)
```

3) Develop user defined function to calculate the correlation coefficient for two numerical attributes and should print these two attributes are correlated (positive / negative) or not. Calculate the covariance.

Data:

age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

Draw the boxplots for age and %fat. Draw the scatter plot on these two variables.

Code and output:

```
import pandas as pd
import numpy as np
age = [23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61]
fat = [9.5,26.5,7.8,17.8,31.4,25.9,27.4,27.2,31.2,34.6,42.5,28.8,33.4,3
0.2,34.1,32.9,41.2,35.7]

data = {'age':age,'fat':fat}
df = pd.DataFrame(data)
n = len(df)

def corcoef(data, N):
  a = int(input("Select your first column number u want to correlate: "
))
  b = int(input("Select your second column number u want to correlate:"
))
  a,b = a-1,b-1
  x = data.iloc[:,a].sum()
  y = data.iloc[:,b].sum()
  xy = (data.iloc[:,a]* data.iloc[:,b]).sum()
  x2 = (data.iloc[:,a]**2).sum()
  y2 = (data.iloc[:,b]**2).sum()
  std_x = df.iloc[:,a].std()
  std_y = df.iloc[:,b].std()
  r = (N*xy - (x*y)) / np.sqrt((N*x2-(x**2))*2(N*y2-(y**2)))
  cov = r*(std_x)*(std_y)

  print("The correlation coefficient is: ",r)
  print("The covariance is: ",cov)

  if r>0:
     print("The 2 variables are positively correlated")
  else:
     print("The 2 variables are negatively correlated")

corcoef(df,n)
```

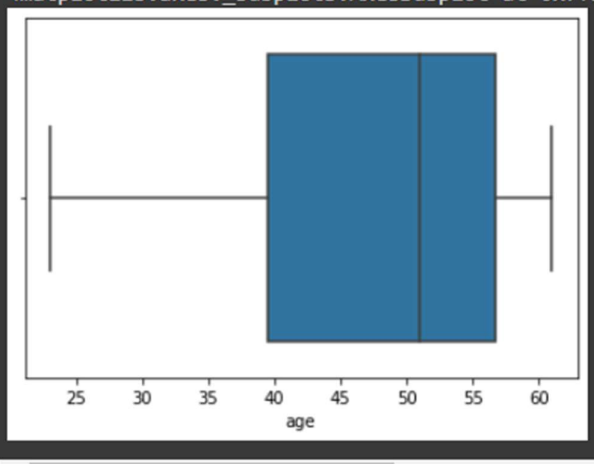
```python
import pandas as pd
import numpy as np
age = [23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61]
fat = [9.5,26.5,7.8,17.8,31.4,25.9,27.4,27.2,31.2,34.6,42.5,28.8,33.4,30.2,34.1,32.9,41.2,35.7]

data = {'age':age,'fat':fat}
df = pd.DataFrame(data)
n = len(df)

def corcoef(data, N):
  a = int(input("Select your first column number u want to correlate: "))
  b = int(input("Select your second column number u want to correlate:"))
  a,b = a-1,b-1
  x = data.iloc[:,a].sum()
  y = data.iloc[:,b].sum()
  xy = (data.iloc[:,a]* data.iloc[:,b]).sum()
  x2 = (data.iloc[:,a]**2).sum()
  y2 = (data.iloc[:,b]**2).sum()
  std_x = df.iloc[:,a].std()
  std_y = df.iloc[:,b].std()
  r = (N*xy - (x*y)) / np.sqrt((N*x2-(x**2))*2(N*y2-(y**2)))
  cov = r*(std_x)*(std_y)

  print("The correlation coefficient is: ",r)
  print("The covariance is: ",cov)

  if r>0:
     print("The 2 variables are positively correlated")
  else:
     print("The 2 variables are negatively correlated")

corcoef(df,n)
```

```
Select your first column number u want to correlate: 1
Select your second column number u want to correlate:2
The correlation coefficient is:  0.8176187964565881
The covariance is:  100.01960784313736
The 2 variables are positively correlated
```

```python
import seaborn as sns
sns.boxplot('age',data=df)
```

```
import seaborn as sns
sns.boxplot('age',data=df)
```

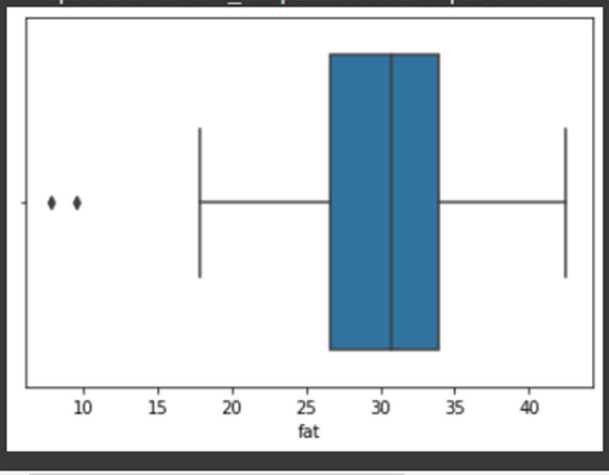
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f18a50c7950>
```




```
sns.boxplot('fat',data=df)
```

```
sns.boxplot('fat',data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f18a505e710>
```



```
sns.scatterplot(data=df,x="age",y="fat",color="red")
```

```
sns.scatterplot(data=df,x="age",y="fat",color="red")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f18a4fc2950>
```