DATA SCIENCE

A PRACTICAL REPORT
ON
DATA SCIENCE

SUBMITTED BY
Mr. RAHUL KEWAT
Roll No: 22006

UNDER THE GUIDANCE OF
Mrs. DIPIKA MANKAR

Submitted in fulfillment of the requirements for qualifying
MSc. IT Part I Semester - I Examination 2022-2023

University of Mumbai
Department of Information Technology

R.D. & S.H National College of Arts, Commerce & S.W.A.
Science College Bandra (West), Mumbai – 400 050.

R. D. & S. H. National & S. W. A. Science College

Bandra (W), Mumbai _ 400050.

Department of Information Technology
M.Sc. (IT _ SEMESTER I)

Certificate

*This is to certify that **Data Science Practicals** performed at R.D & S.H National & S.W.A. Science College by Mr.**Rahul Kewat** holding Seat No. **22006** studying Master of Science in Information Technology Semester – I has been satisfactorily completed as prescribed by the University of Mumbai, during the year 2022 – 2023.*

Subject In-Charge **Coordinator In-Charge** **External Examiner**

College Stamp

INDEX

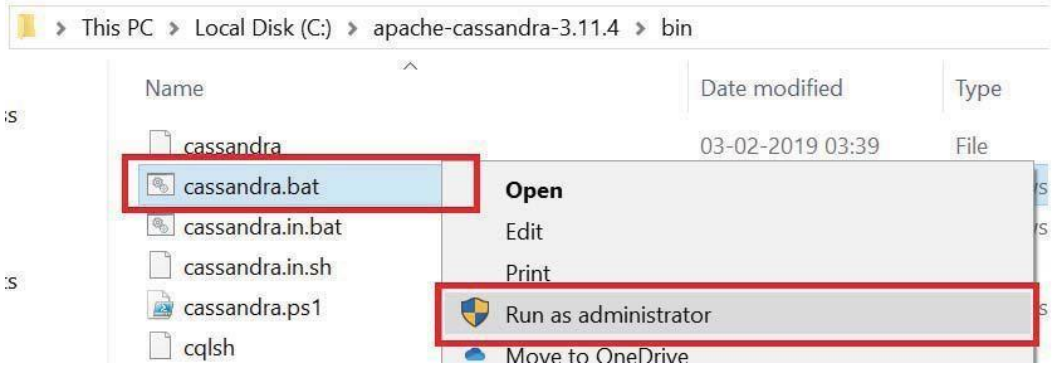
Writeups:

Practical No: - 1

Practical 1:

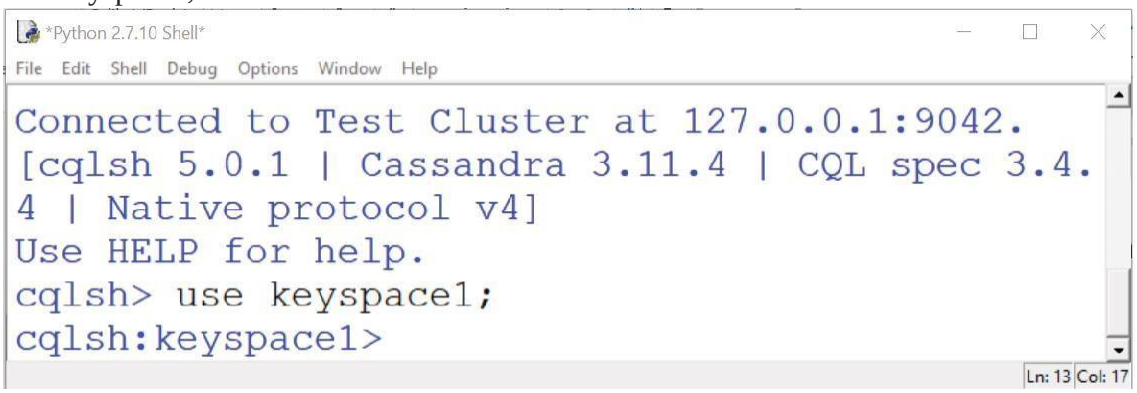
Creating Data Model using Cassandra.

Go to Cassandra directory
C:\apache-cassandra-3.11.4\bin



Run Cassandra.bat file
Open C:\apache-cassandra-3.11.4\bin\cqlsh.py with python 2.7 and run

Creating a Keyspace using Cqlsh
Create keyspace keyspace1 with replication = {'class':'SimpleStratergy', 'replication_factor': 3};
Use keyspace1;



Create table dept (dept_id int PRIMARY KEY, dept_name text, dept_loc text); Create table emp (emp_id int PRIMARY KEY, emp_name text, dept_id int, email text, phone text);

Insert into dept (dept_id, dept_name, dept_loc) values (1001, 'Accounts', 'Mumbai'); Insert into dept (dept_id, dept_name, dept_loc) values (1002, 'Marketing', 'Delhi'); Insert into dept (dept_id, dept_name, dept_loc) values (1003, 'HR', 'Chennai');

Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1001, 'ABCD', 1001, 'abcd@company.com', '1122334455');
Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1002, 'DEFG'

1001, 'defg@company.com', '2233445566'); Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1003, 'GHIJ', 1002, 'ghij@company.com', '3344556677'); Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1004, 'JKLM', 1002, 'jklm@company.com', '4455667788'); Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1005, 'MNOP', 1003, 'mnop@company.com', '5566778899'); Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1006, 'MNOP', 1003, 'mnop@company.com', '5566778844');

```
cqlsh:keyspace1> select * from emp;

 emp_id | dept_id | email             | emp_name | phone
--------+---------+-------------------+----------+------------
   1006 |    1003 | mnop@company.com |     MNOP | 5566778844
   1004 |    1002 | jklm@company.com |     JKLM | 4455667788
   1005 |    1003 | mnop@company.com |     MNOP | 5566778899
   1001 |    1001 | abcd@company.com |     ABCD | 1122334455
   1003 |    1002 | ghij@company.com |     GHIJ | 3344556677
   1002 |    1001 | defg@company.com |     DEFG | 2233445566

(6 rows)
```

```
cqlsh:keyspace1> select * from dept;

 dept_id | dept_loc | dept_name
---------+----------+-----------
    1001 |   Mumbai |  Accounts
    1003 |  Chennai |         HR
    1002 |    Delhi | Marketing

(3 rows)
```

update dept set dept_name='Human Resource' where dept_id=1003;

```
cqlsh:keyspace1> select * from dept;

 dept_id | dept_loc | dept_name
---------+----------+----------------
    1001 |   Mumbai |       Accounts
    1003 |  Chennai | Human Resource
    1002 |    Delhi |      Marketing

(3 rows)
```

```
cqlsh:keyspace1> delete from emp where emp_id=1006;
cqlsh:keyspace1> select * from emp;

 emp_id | dept_id | email             | emp_name | phone
--------+---------+-------------------+----------+------------
   1004 |    1002 | jklm@company.com |     JKLM | 4455667788
   1005 |    1003 | mnop@company.com |     MNOP | 5566778899
   1001 |    1001 | abcd@company.com |     ABCD | 1122334455
   1003 |    1002 | ghij@company.com |     GHIJ | 3344556677
   1002 |    1001 | defg@company.com |     DEFG | 2233445566

(5 rows)
```

Writeups:

Practical No: - 2

Practical 2:

Write Python / R Program to convert from the following formats to HORUS format:

A. Text delimited CSVto HORUS format.

Code:

```
              # Utility Start CSV to HORUS ================================
# Standard Tools
#=============================================================
import pandas as pd
# Input Agreement ===========================================
sInputFileName='Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
# Processing Rules ==========================================
ProcessData=InputData

# Remove columns ISO-2-Code and ISO-3-CODE

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)

# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)

print('Process Data Values ================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ==========================================
OutputData=ProcessData

sOutputFileName='HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)

print('CSV to HORUS - Done')
```

```
# Utility done ================================================
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul>
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/Desktop/Rahul/CSVtoHORUS.p
y
Input Data Values ==================================
                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0                 Afghanistan         AF         AFG        4
1               Aland Islands         AX         ALA      248
2                     Albania         AL         ALB        8
3                     Algeria         DZ         DZA       12
4              American Samoa         AS         ASM       16
..                        ...        ...         ...      ...
242  Wallis and Futuna Islands         WF         WLF      876
243            Western Sahara         EH         ESH      732
244                     Yemen         YE         YEM      887
245                    Zambia         ZM         ZMB      894
246                  Zimbabwe         ZW         ZWE      716

[247 rows x 4 columns]
================================================
Process Data Values ================================
                           CountryName
CountryNumber
716                           Zimbabwe
894                             Zambia
887                              Yemen
732                     Western Sahara
876          Wallis and Futuna Islands
...                                ...
16                     American Samoa
12                            Algeria
8                             Albania
248                     Aland Islands
4                         Afghanistan

[247 rows x 1 columns]
================================================
CSV to HORUS - Done
PS C:\Users\Dell\Desktop\Rahul>
```

B. XML to HORUS Format

Code:

```
import pandas as pd
import xml.etree.ElementTree as ET
#============================================================
def df2xml(data):
    header = data.columns
    root=ET.Element('root')
    for row in range(data.shape[0]):
        entry = ET.SubElement(root,'entry')
        for index in range(data.shape[1]):
            schild=str(header[index])
            child = ET.SubElement(entry, schild)
            if str(data[schild][row]) != 'nan':
                child.text = str(data[schild][row])
            else:
                child.text = 'n/a'
            entry.append(child)
        result = ET.tostring(root)
        return result
#============================================================
def xml2df(xml_data):
    root = ET.XML(xml_data)
    all_records = []
    for i, child in enumerate(root):
        record = {}
        for subchild in child:
            record[subchild.tag] = subchild.text
        all_records.append(record)
    return pd.DataFrame(all_records)
sInputFileName='Country_Code.xml'
InputData = open(sInputFileName).read()
```

```
print('=====================================================')
print('Input Data Values ===================================')
print('=====================================================')
```

```
#print(InputData)
print('=====================================================')
#============================================================
# Processing Rules ==========================================
#============================================================
ProcessDataXML=InputData
# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('=====================================================')
print('Process Data Values ================================')
print('=====================================================')
print(ProcessData)
print('=====================================================')
OutputData=ProcessData
sOutputFileName='HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('XML to HORUS - Done')
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/Desktop/Rahul/XML-HORUS.py
================================================
Input Data Values ==============================
================================================
================================================
================================================
Process Data Values ============================
================================================
                           CountryName
CountryNumber
716                           Zimbabwe
894                             Zambia
887                              Yemen
732                     Western Sahara
876          Wallis and Futuna Islands
...                                ...
16                     American Samoa
12                            Algeria
8                             Albania
248                     Aland Islands
4                         Afghanistan

[247 rows x 1 columns]
================================================
================================================
XML to HORUS - Done
PS C:\Users\Dell\Desktop\Rahul>
```

C. JSON to HORUS

Code:

```
# Utility Start JSON to HORUS ================================
# Standard Tools
#=============================================================
import pandas as pd
# Input Agreement ============================================
sInputFileName='Country_Code.json'
InputData=pd.read_json(sInputFileName,
                       orient='index',
                       encoding="latin-1")
print('Input Data Values ==================================')
print(InputData)
print('=====================================================')
# Processing Rules ===========================================
ProcessData=InputData

# Remove columns ISO-2-Code and ISO-3-CODE

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)

# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)

print('Process Data Values ================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ===========================================
OutputData=ProcessData

sOutputFileName='HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)

print('JSON to HORUS - Done')
# Utility done ===============================================
```

Output:

```
Input data Values ================================================
                      Country ISO-2-CODE ISO-3-Code  ISO-M49
0                 Afghanistan         AF        AFG         4
1               Aland Islands         AX        ALA       248
2                     Albania         AL        ALB         8
3                     Algeria         DZ        DZA        12
4              American Samoa         AS        ASM        16
..                       ...        ...        ...       ...
242  Wallis and Futuna Islands         WF        WLF       876
243            Western Sahara         EH        ESH       732
244                     Yemen         YE        YEM       887
245                    Zambia         ZM        ZMB       894
246                  Zimbabwe         ZW        ZWE       716

[247 rows x 4 columns]
==================================================================
Process Data Values ==============================================
                             CountryName
CountryNumber
716                             Zimbabwe
894                               Zambia
887                                Yemen
732                        Western Sahara
876            Wallis and Futuna Islands
...                                  ...
16                        American Samoa
12                               Algeria
8                                Albania
248                        Aland Islands
4                            Afghanistan

[247 rows x 1 columns]
==================================================================
JSON to HORUS - Done
```

D. MySql Database to HORUS Format

Code:

```
#============================================================
import pandas as pd
import sqlite3 as sq
# Input Agreement ============================================
sInputFileName='C:\\Users\\Dell\\Downloads\\csv\\utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)
print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
# Processing Rules ===========================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ===========================================
OutputData=ProcessData
sOutputFileName='C:\\Users\\Dell\\Downloads\\csv\\HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('Database to HORUS - Done')
# Utility done ===============================================
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul>
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/Desktop/Rahul/Mysql.py
Input Data Values ==================================
     index                 Country ISO-2-CODE ISO-3-Code  ISO-M49
0        0             Afghanistan         AF        AFG        4
1        1           Aland Islands         AX        ALA      248
2        2                 Albania         AL        ALB        8
3        3                 Algeria         DZ        DZA       12
4        4          American Samoa         AS        ASM       16
..     ...                     ...        ...        ...      ...
242    242  Wallis and Futuna Islands         WF        WLF      876
243    243          Western Sahara         EH        ESH      732
244    244                   Yemen         YE        YEM      887
245    245                  Zambia         ZM        ZMB      894
246    246                Zimbabwe         ZW        ZWE      716

[247 rows x 5 columns]
================================================
Process Data Values ================================
             index               CountryName
CountryNumber
716              246                  Zimbabwe
894              245                    Zambia
887              244                     Yemen
732              243            Western Sahara
876              242  Wallis and Futuna Islands
...              ...                       ...
16                 4            American Samoa
12                 3                   Algeria
8                  2                   Albania
248                1             Aland Islands
4                  0               Afghanistan

[247 rows x 2 columns]
================================================
Database to HORUS - Done
PS C:\Users\Dell\Desktop\Rahul>
```

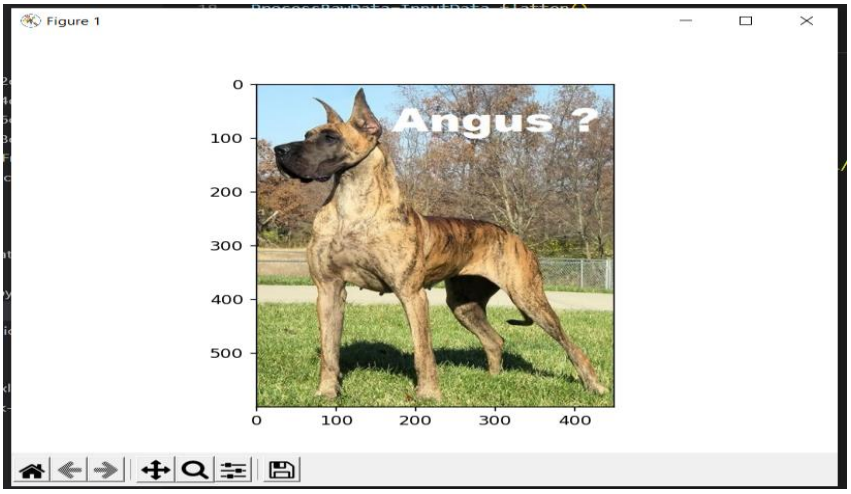
E. Picture (JPEG) to HORUS Format

Code:

```
# Utility Start Picture to HORUS ================================
# Standard Tools
#=============================================================
from matplotlib.pyplot import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Input Agreement ============================================
sInputFileName='C:\\VKHCG\\05-DS\\9999-Data\\Angus (1).jpg'
InputData = imread(sInputFileName)

print('Input Data Values ===================================')
print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])
print('=====================================================')
# Processing Rules ===========================================
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']
# ProcessData.columns=sColumns
ProcessData.index.names =['ID']
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
print('=====================================================')
print('Process Data Values =================================')
print('=====================================================')
plt.imshow(InputData)
plt.show()
print('=====================================================')
# Output Agreement ===========================================
OutputData=ProcessData
print('Storing File')
sOutputFileName='HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Picture to HORUS - Done')
print('=====================================================')
# Utility done ===============================================
```

Output:

```
PS C:\Users\Dell\OneDrive\Documents\ric> & C:/Users/Dell/
ture1.py
Input Data Values ===================================
X:  600
Y:  450
RGBA:  3
=====================================================
Rows:  162000
Columns : 5
=====================================================
Process Data Values =================================
=====================================================
=====================================================
Storing File
=====================================================
Picture to HORUS - Done
=====================================================
PS C:\Users\Dell\OneDrive\Documents\ric>
```

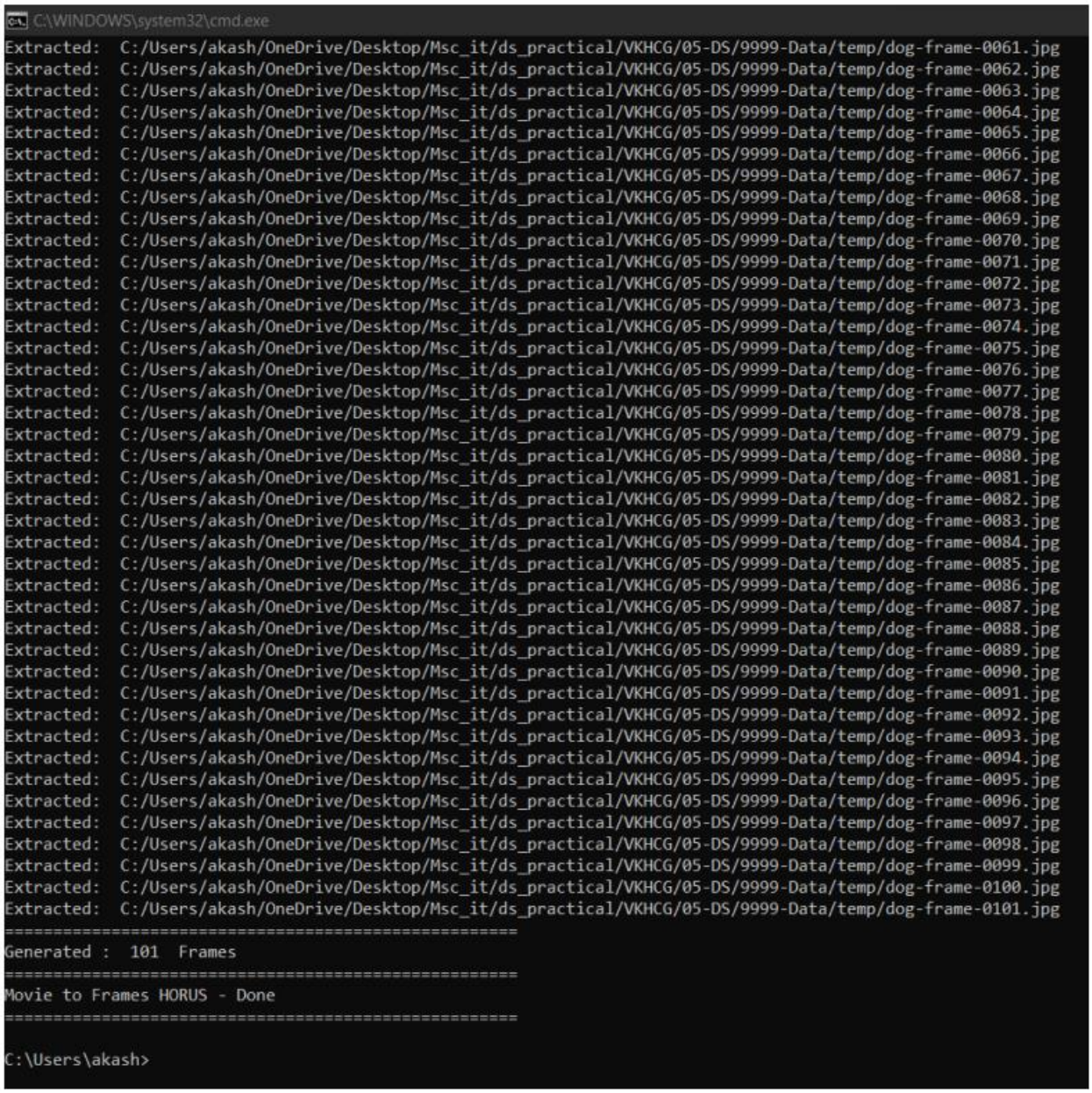


F. Video to HORUS Format

Code:

```
# Standard Tools
#============================================================
import os
 import shutil
 import cv2
 #============================================================
sInputFileName='C:/Users/akash/OneDrive/Desktop/Msc_it/ds_practical/VKHCG/05-D S/9999-Data/dog.mp4'
sDataBaseDir='C:/Users/akash/OneDrive/Desktop/Msc_it/ds_practical/VKHCG/05-DS/ 9999-Data/temp'
if os.path.exists(sDataBaseDir):
shutil.rmtree(sDataBaseDir)
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
print('=====================================================')
print('Start Movie to Frames')
print('=====================================================')
 vidcap = cv2.VideoCapture(sInputFileName)
success,image = vidcap.read()
 count = 0
while success:
success,image = vidcap.read()
sFrame=sDataBaseDir + str('/dog-frame-' + str(format(count, '04d')) + '.jpg')
print('Extracted: ', sFrame)
cv2.imwrite(sFrame, image)
if os.path.getsize(sFrame) == 0:
count += -1
os.remove(sFrame)
print('Removed: ', sFrame)

if cv2.waitKey(10) == 27: # exit if Escape is hit
break if count > 100: # exit
break
count += 1
print('=====================================================')
```

```
print('Generated : ', count, ' Frames')
print('====================================================')
print('Movie to Frames HORUS - Done')
print('====================================================')
# Utility done ============================================
```



G. Frames to Horus

```
# Utility Start Movie to HORUS (Part 2) ======================
# Standard Tools
#=============================================================
from matplotlib.pyplot import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
# Input Agreement ============================================
sDataBaseDir='C:/VKHCG/05-DS/9999-Data/temp'
f=0
for file in os.listdir(sDataBaseDir):
    if file.endswith(".jpg"):
        f += 1
        sInputFileName=os.path.join(sDataBaseDir, file)
        print('Process : ', sInputFileName)

        InputData = imread(sInputFileName)

        print('Input Data Values ===================================')
        print('X: ',InputData.shape[0])
        print('Y: ',InputData.shape[1])
        print('RGBA: ', InputData.shape[2])
        print('=====================================================')
        # Processing Rules ==========================================
        ProcessRawData=InputData.flatten()
        y=InputData.shape[2] + 2
        x=int(ProcessRawData.shape[0]/y)
        ProcessFrameData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
        ProcessFrameData['Frame']=file
        print('=====================================================')
        print('Process Data Values ================================')
        print('=====================================================')
        plt.imshow(InputData)
        plt.show()

        if f == 1:
            ProcessData=ProcessFrameData
        else:
            ProcessData=ProcessData.append(ProcessFrameData)
if f > 0:
    sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha','FrameName']
    ProcessData.columns=sColumns
```

```
    print('=====================================================')
    ProcessFrameData.index.names =['ID']
    print('Rows: ',ProcessData.shape[0])
    print('Columns :',ProcessData.shape[1])
    print('=====================================================')
    # Output Agreement ============================================
    OutputData=ProcessData
    print('Storing File')
    sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Movie-Frame.csv'
    OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Processed ; ', f,' frames')
print('=====================================================')
print('Movie to HORUS - Done')
print('=====================================================')
# Utility done ================================================
```

Output:

```
PS C:\Users\Dell\OneDrive\Documents\ric> & C:/Users/Dell/Appl
me.py
Process :  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0000.jpg
Input Data Values ==================================
X:  360
Y:  640
RGBA:  3
=====================================================
=====================================================
Process Data Values ================================
=====================================================
```

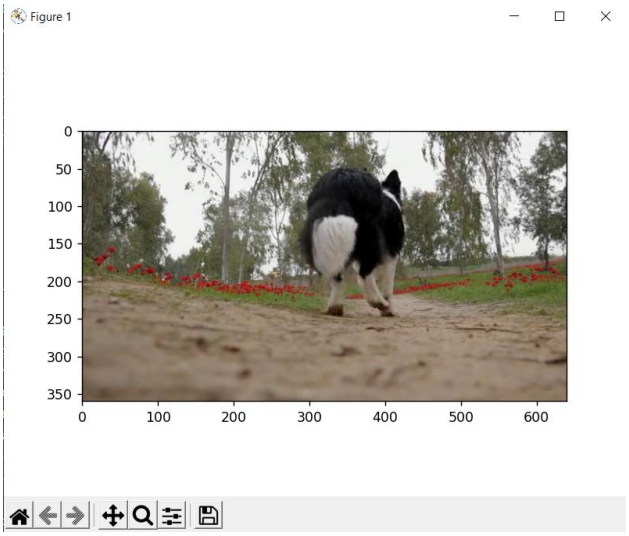


Figure 1
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600

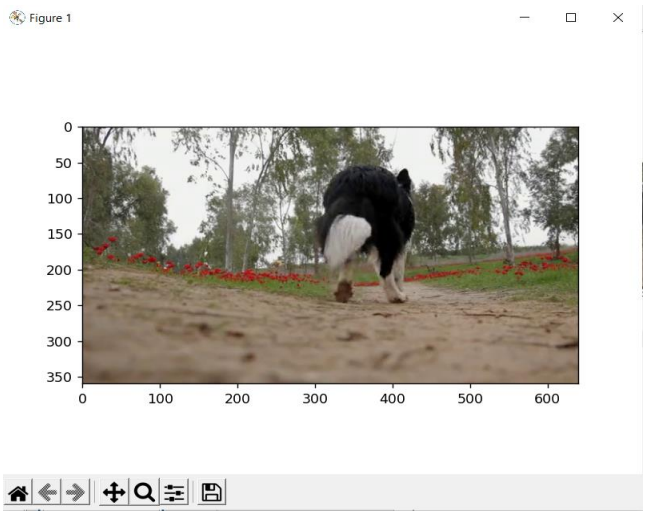


Figure 1
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600


Figure 1
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600

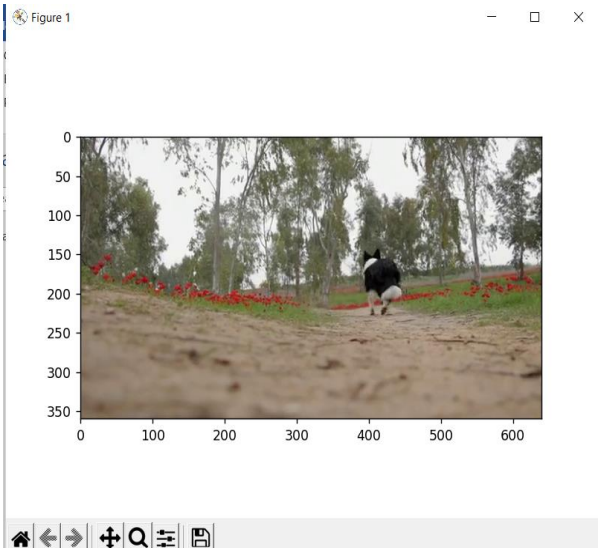


Figure 1
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600

H. Audio to HORUS Format Code:

```
# Utility Start Audio to HORUS ================================
# Standard Tools
#=============================================================
from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#=============================================================
def show_info(aname, a,r):
    print ('----------------')
    print ("Audio:", aname)
    print ('----------------')
    print ("Rate:", r)
    print ('----------------')
    print ("shape:", a.shape)
    print ("dtype:", a.dtype)
    print ("min, max:", a.min(), a.max())
    print ('----------------')
    plot_info(aname, a,r)
#=============================================================
def plot_info(aname, a,r):
    sTitle= 'Signal Wave - '+ aname + ' at ' + str(r) + 'hz'
    plt.title(sTitle)
    sLegend=[]
    for c in range(a.shape[1]):
        sLabel = 'Ch' + str(c+1)
        sLegend=sLegend+[str(c+1)]
        plt.plot(a[:,c], label=sLabel)
    plt.legend(sLegend)
    plt.show()
#=============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'
print('=======================================================')
print('Processing : ', sInputFileName)
print('=======================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
```

```
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("4 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("6 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/8ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("8 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Audio to HORUS - Done')
print('=====================================================')
#============================================================
# Utility done ==============================================
```

```
#============================================================
```

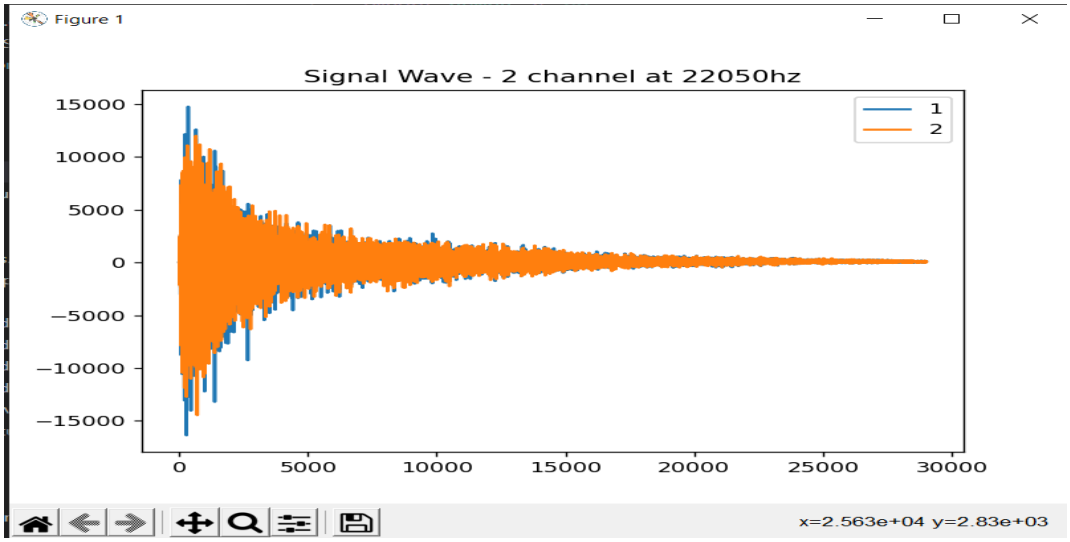
Output:

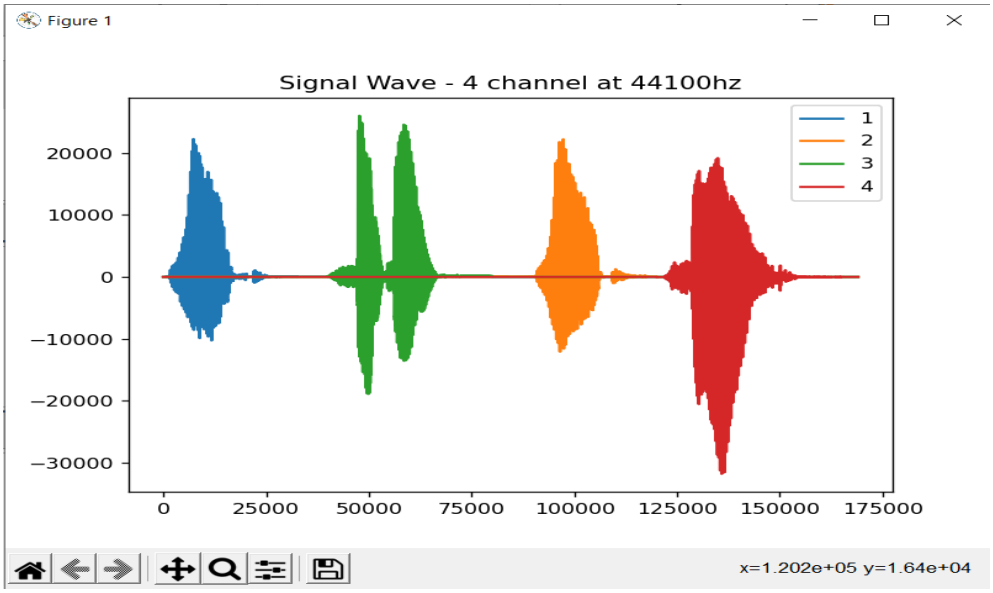
```
PS C:\Users\Dell\OneDrive\Documents\ric> & C:/Users/Dell/AppDa
io.py
====================================================
Processing :  C:/VKHCG/05-DS/9999-Data/2ch-sound.wav
====================================================
---------------
Audio: 2 channel
---------------
Rate: 22050
---------------
shape: (29016, 2)
dtype: int16
min, max: -16384 14767
---------------
```

```
====================================================
Processing :  C:/VKHCG/05-DS/9999-Data/4ch-sound.wav
====================================================
---------------
Audio: 4 channel
---------------
Rate: 44100
---------------
shape: (169031, 4)
dtype: int16
min, max: -31783 26018
---------------
```

```
====================================================
Processing :  C:/VKHCG/05-DS/9999-Data/6ch-sound.wav
====================================================
---------------
Audio: 6 channel
---------------
Rate: 44100
---------------
shape: (257411, 6)
dtype: int16
min, max: -10018 10957
---------------
```

```
==================================================
Processing :  C:/VKHCG/05-DS/9999-Data/8ch-sound.wav
==================================================
---------------
Audio: 8 channel
---------------
Rate: 48000
---------------
shape: (888000, 8)
dtype: int16
min, max: -24859 16303
---------------
```





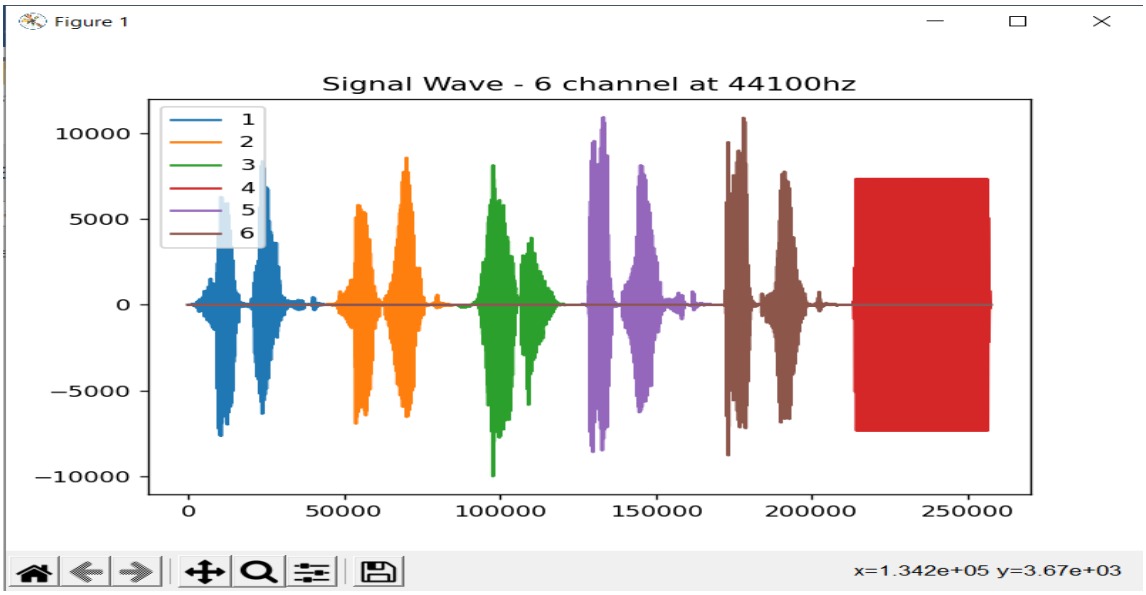


Figure 1
Signal Wave - 6 channel at 44100hz
10000
5000
0
−5000
−10000
0
50000
100000
150000
200000
250000
1
2
3
4
5
6
x=1.342e+05 y=3.67e+03

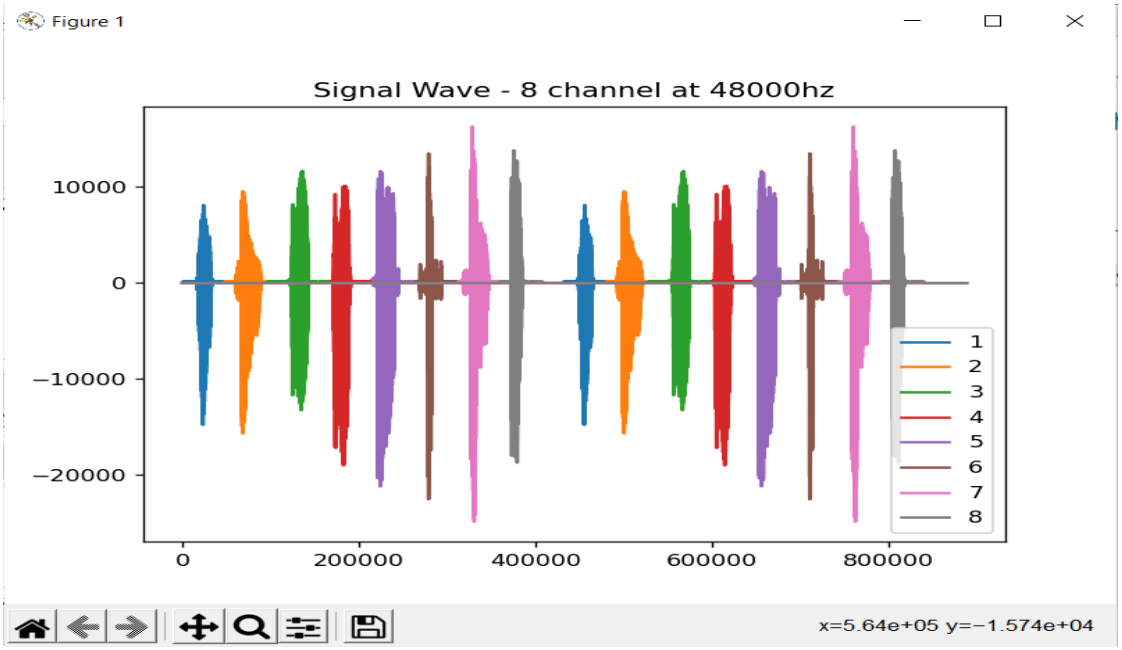


Figure 1
Signal Wave - 8 channel at 48000hz
10000
0
−10000
−20000
0
200000
400000
600000
800000
1
2
3
4
5
6
7
8
x=5.64e+05 y=−1.574e+04

Writeups:

Practical No: - 3

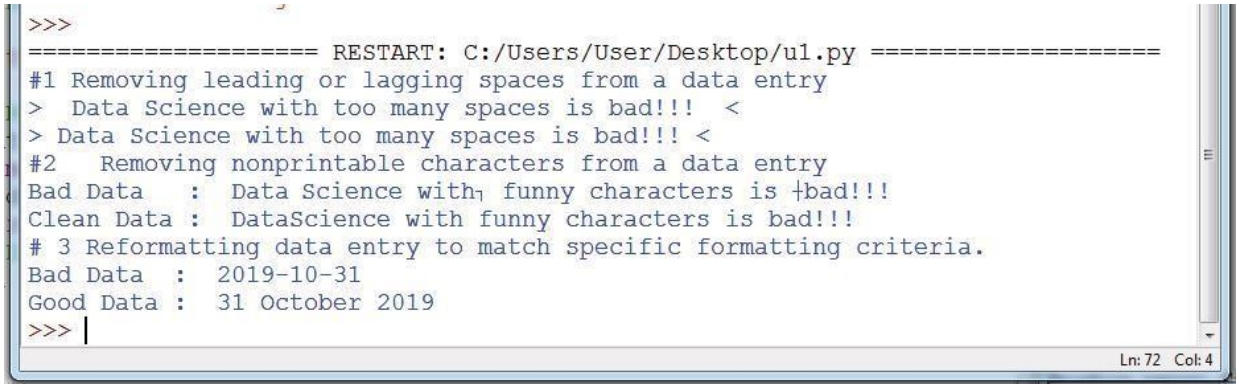
Practical 3: Utilities and Auditing

A. Fixers Utilities:

Fixers enable your solution to take your existing data and fix a specific quality issue.

```
#---------------------------- Program to Demonstrate Fixers utilities --------------------
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print('#1 Removing leading or lagging spaces from a data entry');
baddata = " Data Science with too many spaces is
bad!!! " print('>',baddata,'<')
cleandata=baddata.strip()
print('>',cleandata,'<')

#2 Removing nonprintable characters from a data entry
print('#2 Removing nonprintable characters from a data entry')
printable = set(string.printable)
baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!"
cleandata=''.join(filter(lambda x: x in string.printable,baddata))
print('Bad Data : ',baddata); print('Clean Data : ',cleandata)

# 3 Reformatting data entry to match specific formatting criteria.
# Convert YYYY/MM/DD to DD Month YYYY
print('# 3 Reformatting data entry to match specific formatting criteria.')
baddate = dt.date(2019, 10, 31) baddata=format(baddate,'%Y-%m-%d')
gooddate = dt.datetime.strptime(baddata,'%Y-%m-%d')
gooddata=format(gooddate,'%d %B %Y') print('Bad Data : ',baddata)
print('Good Data : ',gooddata)
```

Output:



B. Data Binning or Bucketing

Binning is a data preprocessing technique used to reduce the effects of minor observation errors. Statistical data binning is a way to group a number of more or less continuous values into a smaller number of "bins."

Code :

```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import scipy.stats as stats

np.random.seed(0)

# example data
mu = 90 # mean of distribution
sigma = 25 # standard deviation of distribution
x = mu + sigma*np.random.randn(5000)
num_bins = 25
fig, ax = plt.subplots()

# the histogram of the data
n, bins, patches = ax.hist(x,num_bins,density = 1)

# add a 'best fit' line

y = stats.norm.pdf(bins, mu, sigma)

# mlab.normpdf(bins, mu, sigma)
ax.plot(bins, y, '--')
ax.set_xlabel('Example Data')
ax.set_ylabel('Probability density')
sTitle = r'Histogram' +str(len(x)) + 'entries into' +str(num_bins)
+'Bins:  $\mu=' + str(mu) + 'str(mu) + $, $\sigma =' + str(sigma) + '$'
ax.set_title(sTitle)
fig.tight_layout()
sPathFig = 'C:\\Users\\Dell\\Desktop\\Rahul\\Histogram.png'
fig.savefig(sPathFig)
plt.show()
```

Output:

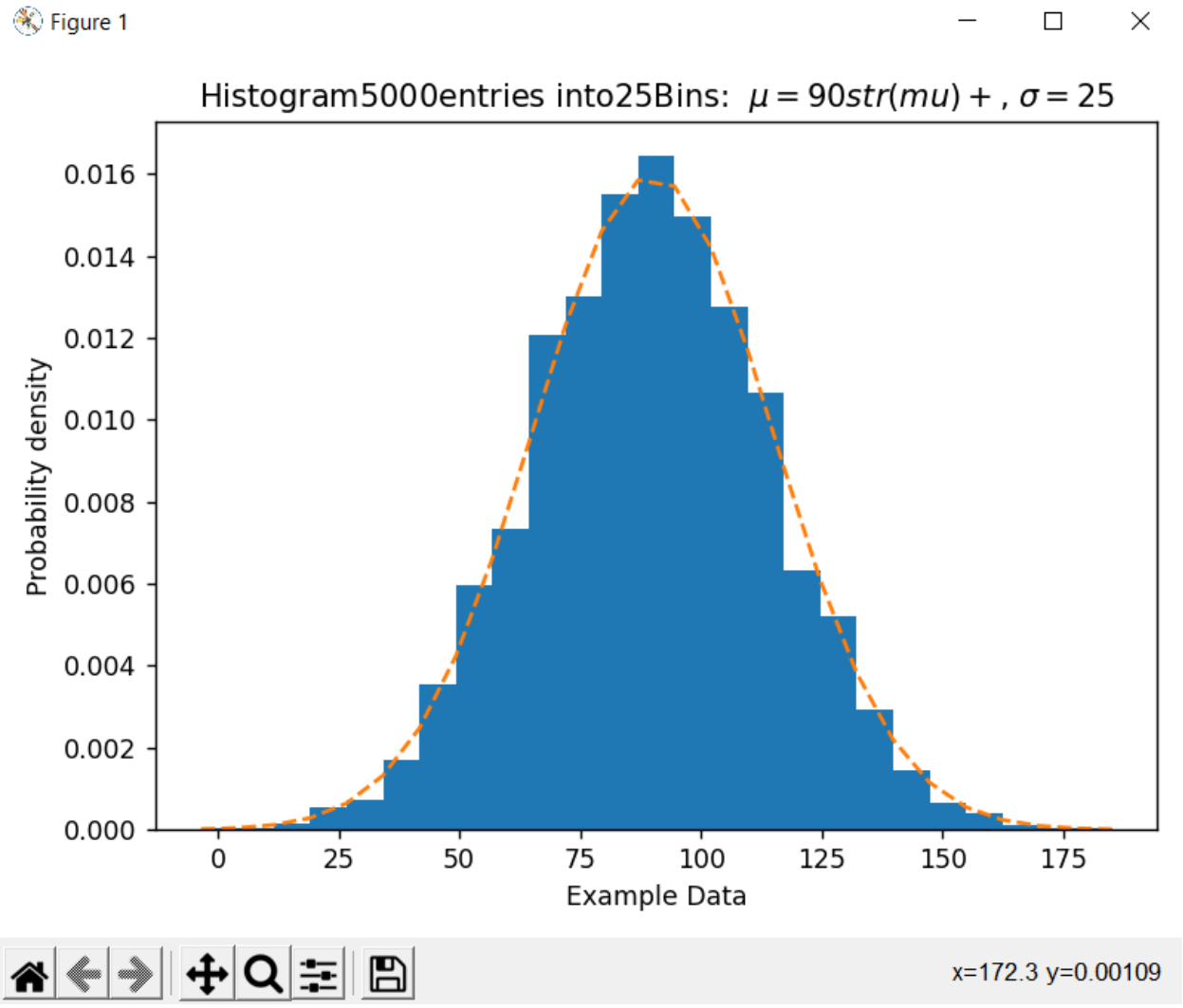


Figure 1
Histogram5000entries into25Bins: $\mu = 90str(mu) + , \sigma = 25$
Probability density
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
25
50
75
100
125
150
175
Example Data
x=172.3 y=0.00109

C. Averaging of Data

The use of averaging of features value enables the reduction of data volumes in a control fashion to improve effective data processing. C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Mean.py

Code:

```
import pandas as pd
#################################################################
InputFileName='C:\\Users\\Dell\\Desktop\\Rahul\\IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'

print('#################################')
IP_DATA_ALL=pd.read_csv(header=0,low_memory=False, usecols=['Country','Place
Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
AllData=IP_DATA_ALL[['Country', 'Place_Name','Latitude']]
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
print(MeanData)
```

Output:

```
= RESTART: C:\Users\Dell\AppData\Local\Programs
#################################
     Country Place_Name  Latitude
0         US   New York   40.7528
1         US   New York   40.7528
2         US   New York   40.7528
3         US   New York   40.7528
4         US   New York   40.7528
...      ...        ...       ...
3557      DE     Munich   48.0915
3558      DE     Munich   48.1833
3559      DE     Munich   48.1000
3560      DE     Munich   48.1480
3561      DE     Munich   48.1480

[3562 rows x 3 columns]
Country  Place_Name
DE       Munich        48.143223
GB       London        51.509406
US       New York      40.747044
Name: Latitude, dtype: float64
```

D. Outlier Detection

Outliers are data that is so different from the rest of the data in the data set that it may be caused by an error in the data source. There is a technique called outlier detection that, with good data science, will identify these outliers.

C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py

Code:

```
#-*- coding: utf-8 -*-
###############################################################
import pandas as pd
###############################################################

InputFileName='C:\\Users\\Dell\\Downloads\\csv\\xmlhorus\\IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('##############################')

IP_DATA_ALL=pd.read_csv(InputFileName,low_memory=False,usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data')
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) &
(AllData.Latitude<=UpperBound)]
print(OutliersNot)
```

Output:

```
= RESTART: C:\Users\Dell\AppData\Local\Programs\Python\Python310\CSV to HORUS\utlity.p
############################
All Data
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
...      ...        ...       ...
3434      GB     London   51.5092
3435      GB     London   51.5092
3436      GB     London   51.5163
3437      GB     London   51.5085
3438      GB     London   51.5136

[1502 rows x 3 columns]
Outliers
Higher than  51.512635507867415
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
1916      GB     London   51.5491
1919      GB     London   51.5161
1920      GB     London   51.5198
1921      GB     London   51.5198
1923      GB     London   51.5237
1924      GB     London   51.5237
1925      GB     London   51.5237
1926      GB     London   51.5237
1927      GB     London   51.5232
3436      GB     London   51.5163
3438      GB     London   51.5136
Lower than  51.506176875621264
     Country Place_Name  Latitude
1915      GB     London   51.4739
Not Outliers
     Country Place_Name  Latitude
1917      GB     London   51.5085
1918      GB     London   51.5085
1922      GB     London   51.5085
1928      GB     London   51.5085
1929      GB     London   51.5085
...      ...        ...       ...
3432      GB     London   51.5092
3433      GB     London   51.5092
3434      GB     London   51.5092
3435      GB     London   51.5092
3437      GB     London   51.5085

[1485 rows x 3 columns]
```

E. Logging

Write a Python / R program for basic logging in data science.

```
import shutil
import sys
import os
import logging
import uuid
import shutil
import time

Base='C:\VKHCG'

sCompanies = ['01--Vermeulen','02-Krennwallner','03-Hillman','04-Clark']

sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-
Organise','06-Report']

sLevels=['debug','info','warning','error']

for sCompany in sCompanies:
    sFileDir=Base + '/' + sCompany
    if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
    for sLayer in sLayers:
        log = logging.getLogger() # root logger

        for hdlr in log.handlers[:]: # remove all old handlers
            log.removeHandler(hdlr)

        sFileDir = Base + '/' + sCompany + '/' + sLayer + '/Logging'
        if os.path.exists(sFileDir):
            shutil.rmtree(sFileDir)
        time.sleep(2)
        if not os.path.exists(sFileDir):
            os.makedirs(sFileDir)
        skey = str(uuid.uuid4())

        sLogFile=Base + '/' + sCompany + '/' + sLayer + '/Logging/Logging_'+
skey + '.log'
        print('Set up:',sLogFile)

        logging.basicConfig(level=logging.DEBUG,
                            format='%(asctime)s %(name)-12s %(levelname)-8s
%(message)s',
```

```
                            datefmt='%m-%d %H:%M',
                            filename=sLogFile,
                            filemode='w')

        sys.stderr
        console = logging.StreamHandler()
        console.setLevel(logging.INFO)

        formatter = logging.Formatter('%(name)-12s: %(levelname)-8s
%(message)s')

        console.setFormatter(formatter)

        logging.getLogger('').addHandler(console)

        logging.info('Practical Data Science is fun!.')

        for sLevel in sLevels:
            sApp='Apllication-'+ sCompany + '-' + sLayer + '-' + sLevel
            logger = logging.getLogger(sApp)
            if sLevel == 'debug':
                logger.debug('Practical Data Science logged a debugging
message.')
            if sLevel == 'info':
                logger.info('Practical Data Science logged information
message.')
            if sLevel == 'warning':
                logger.warning('Practical Data Science logged a warning
message.')
            if sLevel == 'error':
                logger.error('Practical Data Science logged an error message.')
```

Output:

```
PS C:\Users\Dell\OneDrive\Documents\ric> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/C
ging.py
Set up: C:\VKHCG/01--Vermeulen/01-Retrieve/Logging/Logging_b5f72f67-16cf-4954-9cbe-1e7fe835bdc8.log
root        : INFO     Practical Data Science is fun!.
Apllication-01--Vermeulen-01-Retrieve-info: INFO     Practical Data Science logged information message.
Apllication-01--Vermeulen-01-Retrieve-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01--Vermeulen-01-Retrieve-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/01--Vermeulen/02-Assess/Logging/Logging_b98c83e2-1629-472b-82ac-a13b01743586.log
root        : INFO     Practical Data Science is fun!.
Apllication-01--Vermeulen-02-Assess-info: INFO     Practical Data Science logged information message.
Apllication-01--Vermeulen-02-Assess-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01--Vermeulen-02-Assess-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/01--Vermeulen/03-Process/Logging/Logging_46576e8e-0e2c-4167-ae19-4425c5e96207.log
root        : INFO     Practical Data Science is fun!.
Apllication-01--Vermeulen-03-Process-info: INFO     Practical Data Science logged information message.
Apllication-01--Vermeulen-03-Process-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01--Vermeulen-03-Process-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/01--Vermeulen/04-Transform/Logging/Logging_277c3d9a-cfbc-4274-9acc-b810f7d08c0d.log
root        : INFO     Practical Data Science is fun!.
Apllication-01--Vermeulen-04-Transform-info: INFO     Practical Data Science logged information message.
Apllication-01--Vermeulen-04-Transform-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01--Vermeulen-04-Transform-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/01--Vermeulen/05-Organise/Logging/Logging_d20abf14-af31-4c47-a4db-0776ad2e2ae1.log
root        : INFO     Practical Data Science is fun!.
Apllication-01--Vermeulen-05-Organise-info: INFO     Practical Data Science logged information message.
Apllication-01--Vermeulen-05-Organise-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01--Vermeulen-05-Organise-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/01--Vermeulen/06-Report/Logging/Logging_c6029520-2a56-406d-9bda-63311d890a4c.log
root        : INFO     Practical Data Science is fun!.
Apllication-01--Vermeulen-06-Report-info: INFO     Practical Data Science logged information message.
Apllication-01--Vermeulen-06-Report-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01--Vermeulen-06-Report-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/02-Krennwallner/01-Retrieve/Logging/Logging_4ae43e52-5bd2-4a58-b18f-17f38313b125.log
root        : INFO     Practical Data Science is fun!.
Apllication-02-Krennwallner-01-Retrieve-info: INFO     Practical Data Science logged information message.
Apllication-02-Krennwallner-01-Retrieve-warning: WARNING  Practical Data Science logged a warning message.
Apllication-02-Krennwallner-01-Retrieve-error: ERROR    Practical Data Science logged an error message.
```

```
Set up: C:\VKHCG/02-Krennwallner/02-Assess/Logging/Logging_4c69bbf9-e96a-40e6-a5f3-cb18188019c0.log
root        : INFO     Practical Data Science is fun!.
Apllication-02-Krennwallner-02-Assess-info: INFO     Practical Data Science logged information message.
Apllication-02-Krennwallner-02-Assess-warning: WARNING  Practical Data Science logged a warning message.
Apllication-02-Krennwallner-02-Assess-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/02-Krennwallner/03-Process/Logging/Logging_d644ebec-9e7c-43b3-8acb-ed1b78b49ee8.log
root        : INFO     Practical Data Science is fun!.
Apllication-02-Krennwallner-03-Process-info: INFO     Practical Data Science logged information message.
Apllication-02-Krennwallner-03-Process-warning: WARNING  Practical Data Science logged a warning message.
Apllication-02-Krennwallner-03-Process-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/02-Krennwallner/04-Transform/Logging/Logging_4d5d89a1-9a35-44f9-8656-0a4e3cb8e03b.log
root        : INFO     Practical Data Science is fun!.
Apllication-02-Krennwallner-04-Transform-info: INFO     Practical Data Science logged information message.
Apllication-02-Krennwallner-04-Transform-warning: WARNING  Practical Data Science logged a warning message.
Apllication-02-Krennwallner-04-Transform-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/02-Krennwallner/05-Organise/Logging/Logging_f38a14e9-cb9b-4fdf-8c39-4c3a10edbd93.log
root        : INFO     Practical Data Science is fun!.
Apllication-02-Krennwallner-05-Organise-info: INFO     Practical Data Science logged information message.
Apllication-02-Krennwallner-05-Organise-warning: WARNING  Practical Data Science logged a warning message.
Apllication-02-Krennwallner-05-Organise-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/02-Krennwallner/06-Report/Logging/Logging_c3860474-823b-4255-b646-3ee17cc9e599.log
root        : INFO     Practical Data Science is fun!.
Apllication-02-Krennwallner-06-Report-info: INFO     Practical Data Science logged information message.
Apllication-02-Krennwallner-06-Report-warning: WARNING  Practical Data Science logged a warning message.
Apllication-02-Krennwallner-06-Report-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/03-Hillman/01-Retrieve/Logging/Logging_4e0d3a8c-8e20-4ce4-aeea-2d508e401c3f.log
root        : INFO     Practical Data Science is fun!.
```

```
Apllication-03-Hillman-05-Organise-info: INFO     Practical Data Science logged information message.
Apllication-03-Hillman-05-Organise-warning: WARNING  Practical Data Science logged a warning message.
Apllication-03-Hillman-05-Organise-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/03-Hillman/06-Report/Logging/Logging_778305fa-2185-445f-b07e-4b963f3da461.log
root        : INFO     Practical Data Science is fun!.
Apllication-03-Hillman-06-Report-info: INFO     Practical Data Science logged information message.
Apllication-03-Hillman-06-Report-warning: WARNING  Practical Data Science logged a warning message.
Apllication-03-Hillman-06-Report-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/04-Clark/01-Retrieve/Logging/Logging_e6e95d53-3d26-4e89-b0ed-ef24ab7dc61a.log
root        : INFO     Practical Data Science is fun!.
Apllication-04-Clark-01-Retrieve-info: INFO     Practical Data Science logged information message.
Apllication-04-Clark-01-Retrieve-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-01-Retrieve-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/04-Clark/02-Assess/Logging/Logging_07def416-a141-4c17-aa65-f4b1bfa17807.log
root        : INFO     Practical Data Science is fun!.
Apllication-04-Clark-02-Assess-info: INFO     Practical Data Science logged information message.
Apllication-04-Clark-02-Assess-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-02-Assess-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/04-Clark/03-Process/Logging/Logging_c7cf6e17-180a-4501-b9a6-e52bf9483860.log
root        : INFO     Practical Data Science is fun!.
Apllication-04-Clark-03-Process-info: INFO     Practical Data Science logged information message.
Apllication-04-Clark-03-Process-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-03-Process-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/04-Clark/04-Transform/Logging/Logging_6cb7097e-b870-4411-9d88-33bc58d1ec2e.log
root        : INFO     Practical Data Science is fun!.
Apllication-04-Clark-04-Transform-info: INFO     Practical Data Science logged information message.
Apllication-04-Clark-04-Transform-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-04-Transform-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/04-Clark/05-Organise/Logging/Logging_2b1bfa2a-b708-4a81-9a71-52457fe0a47d.log
root        : INFO     Practical Data Science is fun!.
Apllication-04-Clark-05-Organise-info: INFO     Practical Data Science logged information message.
Apllication-04-Clark-05-Organise-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-05-Organise-error: ERROR    Practical Data Science logged an error message.
Set up: C:\VKHCG/04-Clark/06-Report/Logging/Logging_f44c1fb9-aa38-4146-9e16-6974a5a1dff6.log
root        : INFO     Practical Data Science is fun!.
Apllication-04-Clark-06-Report-info: INFO     Practical Data Science logged information message.
Apllication-04-Clark-06-Report-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-06-Report-error: ERROR    Practical Data Science logged an error message.
```

Writeups:

Practical No: - 4

A. Program to retrieve different attributes of data.

C:\ VKHCG\01-Vermeulen\01-Retrieve\Retrive_IP_DATA_ALL.py###

```
import sys
import os
import pandas as pd
Base='C:\\VKHCG'


sFileName='C:\\Users\\Dell\\Downloads\\csv\\xmlhorus\\IP_DATA_CORE.csv'
print('Loading:',sFileName)

IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,encoding="latin-1")

sFileDir=Base+'//01-Vermeulen//01-Retrieve//01-EDS//02-python'

if not os.path.exists(sFileDir):

    print('Row:',IP_DATA_ALL.shape[0])
    print('Columns:',IP_DATA_ALL.shape[1])
    print('###Raw Data Set ##############################################')
for i in range(0,len(IP_DATA_ALL.columns)):
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
    print('###Fixed Data Set ####################################')
    IP_DATA_ALL_FIX=IP_DATA_ALL
for i in range(0,len(IP_DATA_ALL.columns)):
    cNameOld=IP_DATA_ALL_FIX.columns[i]+''
    cNameNew=cNameOld.strip().replace("",".")
    IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i])
    print(IP_DATA_ALL_FIX.head())
    print('Fixed Data Set with ID')
    IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
    IP_DATA_ALL_with_ID.index.names=['RowID']
    print(IP_DATA_ALL_with_ID.head())
    sFileName2=sFileDir+'Retrieve_IP_DATA.csv'
    IP_DATA_ALL_with_ID.to_csv(sFileName2,index=True,encoding="latin-1")
    print('###Done!!################')
```

```
Loading: C:\Users\Dell\Downloads\csv\xmlhorus\IP_DATA_CORE.csv
ID <class 'str'>
###Fixed Data Set ####################################
Country <class 'str'>
###Fixed Data Set ####################################
Place Name <class 'str'>
###Fixed Data Set ####################################
Post Code <class 'str'>
###Fixed Data Set ####################################
Latitude <class 'str'>
###Fixed Data Set ####################################
Longitude <class 'str'>
###Fixed Data Set ####################################
First IP Number <class 'str'>
###Fixed Data Set ####################################
Last IP Number <class 'str'>
###Fixed Data Set ####################################
   ID Country Place Name  ... Longitude  First IP Number  Last IP Number
0   1      US   New York  ...  -73.9725        204276480       204276735
1   2      US   New York  ...  -73.9725        301984864       301985791
2   3      US   New York  ...  -73.9725        404678736       404679039
3   4      US   New York  ...  -73.9725        411592704       411592959
4   5      US   New York  ...  -73.9725        416784384       416784639

[5 rows x 8 columns]
Fixed Data Set with ID
       ID Country Place Name  ... Longitude  First IP Number  Last IP Number
RowID                         ...
0       1      US   New York  ...  -73.9725        204276480       204276735
1       2      US   New York  ...  -73.9725        301984864       301985791
2       3      US   New York  ...  -73.9725        404678736       404679039
3       4      US   New York  ...  -73.9725        411592704       411592959
4       5      US   New York  ...  -73.9725        416784384       416784639

[5 rows x 8 columns]
###Done!!################
```

B. Loading IP_DATA_ALL:

This data set contains all the IP address allocations in the world. It will help you to locate your customers when interacting with them online.

Create a new Python script file and save it as Retrieve-IP_DATA_ALL.py in directory C:\VKHCG\01-Vermeulen\01-Retrieve.

##############Retrieve-IP_DATA_ALL.py#######################

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

print('Rows:', IP_DATA_ALL.shape[0])
print('Columns:', IP_DATA_ALL.shape[1])
print('### Raw Data Set #####################################')
for i in range(0,len(IP_DATA_ALL.columns)):
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print('### Fixed Data Set #####################################')
IP_DATA_ALL_FIX=IP_DATA_ALL
for i in range(0,len(IP_DATA_ALL.columns)):
    cNameOld=IP_DATA_ALL_FIX.columns[i] + '      '
    cNameNew=cNameOld.strip().replace(" ", ".")
    IP_DATA_ALL_FIX.columns.values[i] = cNameNew
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
################################################################
#print(IP_DATA_ALL_FIX.head())
################################################################
print('Fixed Data Set with ID')
IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
```

```
IP_DATA_ALL_with_ID.index.names = ['RowID']
#print(IP_DATA_ALL_with_ID.head())

sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")

########################################################################
print('### Done!! ############################################')
########################################################################
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/Desktop/Rahul/loding.py
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 2
Columns: 1
### Raw Data Set ##################################
version https://git-lfs.github.com/spec/v1 <class 'str'>
### Fixed Data Set ################################
version.https://git-lfs.github.com/spec/v1 <class 'str'>
Fixed Data Set with ID
### Done!! ########################################
PS C:\Users\Dell\Desktop\Rahul>
```

Writeups:

Practical No: - 5

Practical 5:

Assessing Data

Assess Superstep

Data quality refers to the condition of a set of qualitative or quantitative variables. Dataquality is a multidimensional measurement of the acceptability of specific data sets. Inbusiness, data quality is measured to determine whether data can be used as a basis forreliable intelligence extraction for supporting organizational decisions. Data profiling involves observing in your data sources all the viewpoints that theinformation offers. The main goal is to determine if individual viewpoints are accurateand complete. The Assess superstep determines what additional processing to apply tothe entries that are noncompliant. Errors

Typically, one of four things can be done with an error to the data.

1. Accept the Error
2. Reject the Error
3. Correct the Error
4. Create a Default Value

A. Perform error management on the given data using pandas package.

Python pandas package enables several automatic error-management features.

File Location: C:\VKHCG\01-Vermeulen\02-Assess

Missing Values in Pandas:

i. Drop the Columns Where All Elements Are Missing Values

	A	B	C	D	E	F	G	H
1	ID	FieldA	FieldB	FieldC	FieldD	FieldE	FieldF	FieldG
2	1	Good	Better	Best	1024		10241	1
3	2	Good		Best	512		5121	2
4	3	Good	Better		256		256	3
5	4	Good	Better	Best			211	4
6	5	Good	Better		64		6411	5
7	6	Good		Best	32		32	6
8	7		Better	Best	16		1611	7
9	8			Best	8		8111	8
10	9				4		41	9
11	10	A	B	C	2		21111	10
12								11
13	10	Good	Better	Best	1024		102411	12
14	10	Good		Best	512		512	13
15	10	Good	Better		256		1256	14
16	10	Good	Better	Best				15
17	10	Good	Better		64		164	16
18	10	Good		Best	32		322	17
19	10		Better	Best	16		163	18
20	10			Best	8		844	19
21	10				4		4555	20
22	10	A	B	C	2		111	21

Code :

```
import sys
import os
import pandas as pd
#################################################################
Base='C:/VKHCG'
#################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-01.csv'
Company='01-Vermeulen'
#################################################################
Base='C:/VKHCG'
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
### Import Warehouse
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('################################')
print('## Raw Data Values')
print('################################')
print(RawData)
print('################################')
print('## Data Profile')
print('################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('################################')
#################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
#################################################################
TestData=RawData.dropna(axis=1, how='all')
#################################################################
print('################################')
```

```
print('## Test Data Values')
print('################################')
print(TestData)
print('################################')
print('## Data Profile')
print('################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('################################')
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#####################################################################
print('################################')
print('### Done!! ####################')
print('################################')
#####################################################################
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
################################
## Raw Data Values
################################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
################################
```

```
################################
Rows : 21
Columns : 8
################################
################################
## Test Data Values
################################
      ID FieldA  FieldB FieldC  FieldD    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0   10241.0       1
1    2.0   Good     NaN   Best   512.0    5121.0       2
2    3.0   Good  Better    NaN   256.0     256.0       3
3    4.0   Good  Better   Best     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0    6411.0       5
5    6.0   Good     NaN   Best    32.0      32.0       6
6    7.0    NaN  Better   Best    16.0    1611.0       7
7    8.0    NaN     NaN   Best     8.0    8111.0       8
8    9.0    NaN     NaN    NaN     4.0      41.0       9
9   10.0      A       B      C     2.0   21111.0      10
10   NaN    NaN     NaN    NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0  102411.0      12
12  10.0   Good     NaN   Best   512.0     512.0      13
13  10.0   Good  Better    NaN   256.0    1256.0      14
14  10.0   Good  Better   Best     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     164.0      16
16  10.0   Good     NaN   Best    32.0     322.0      17
17  10.0    NaN  Better   Best    16.0     163.0      18
18  10.0    NaN     NaN   Best     8.0     844.0      19
19  10.0    NaN     NaN    NaN     4.0    4555.0      20
20  10.0      A       B      C     2.0     111.0      21
################################
## Data Profile
################################
Rows : 21
Columns : 7
################################
################################
### Done!! ####################
################################
```

ii. Drop the Columns Where Any of the Elements Is Missing Values

Code :

```
import sys
import os
import pandas as pd
Base='C:/VKHCG'
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-02.csv'
Company='01-Vermeulen'
################################################################
Base='C:/VKHCG'
################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
################################################################
### Import Warehouse
################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('################################')
print('## Raw Data Values')
print('################################')
print(RawData)
print('################################')
print('## Data Profile')
print('################################')
print('## Data Profile')
print('################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('################################')
################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
################################################################
```

```
TestData=RawData.dropna(axis=1, how='any')
#####################################################################
print('################################')
print('## Test Data Values')
print('################################')
print(TestData)
print('################################')
print('## Data Profile')
print('################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('################################')
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#####################################################################
print('################################')
print('### Done!! ####################')
print('################################')
#####################################################################
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Pro
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
################################
## Raw Data Values
################################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
```

```
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
################################
## Data Profile
################################
## Data Profile
################################
Rows : 21
Columns : 8
################################
################################
## Test Data Values
```

```
   ######################################
       FieldG
   0        1
   1        2
   2        3
   3        4
   4        5
   5        6
   6        7
   7        8
   8        9
   9       10
   10      11
   11      12
   12      13
   13      14
   14      15
   15      16
   16      17
   17      18
   18      19
   19      20
   20      21
   ######################################
   ## Data Profile
   ######################################
   Rows : 21
   Columns : 1
   ######################################
   ######################################
   ### Done!! ###########################
   ######################################
```

iii. Keep Only the Rows That Contain a Maximum of Two Missing Values

Code :

```
import sys
import os
import pandas as pd
#################################################################
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-03.csv'
Company='01-Vermeulen'
Base='C:/VKHCG'
#################################################################
print('################################')
print('Working Base :',Base, ' using Windows ~~~~')
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
### Import Warehouse
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('################################')
print('## Raw Data Values')
print('################################')
print(RawData)
print('################################')
print('## Data Profile')
print('################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('################################')
#################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
#################################################################
TestData=RawData.dropna(thresh=2)
print('################################')
print('## Test Data Values')
```

```
print('################################')
print(TestData)
print('################################')
print('## Data Profile')
print('################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('################################')
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
######################################################################
print('################################')
print('### Done!! ####################')
print('################################')
######################################################################
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Progr
################################
Working Base : C:/VKHCG  using Windows ~~~~
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
################################
## Raw Data Values
################################
     ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0   1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1   2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2   3.0   Good  Better    NaN   256.0     NaN     256.0       3
3   4.0   Good  Better   Best     NaN     NaN     211.0       4
4   5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5   6.0   Good     NaN   Best    32.0     NaN      32.0       6
6   7.0    NaN  Better   Best    16.0     NaN    1611.0       7
```

```
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
################################
## Data Profile
################################
Rows : 21
Columns : 8
################################
################################
## Test Data Values
################################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
```

```
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
##############################
## Data Profile
##############################
Rows : 20
Columns : 8
##############################
##############################
### Done!! ###################
##############################
```

B. Write Python / R program to create the network routing diagram from the given data onrouters.

########## Assess-Network-Routing-Company.py ####################

```
import sys
import os
import pandas as pd

pd.options.mode.chained_assignment = None

Base='C:/VKHCG'

print('################################')
print('Working Base :',Base, ' using Windows')
print('################################')

sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sInputFileName3='01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv'
sOutputFileName='Assess-Network-Routing-Company.csv'
Company='01-Vermeulen'

#Import Country Data

sFileName=Base + '/' + Company + '/' + sInputFileName1

print('################################')
print('Loading :',sFileName)
print('################################')

CountryData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
print('################################')

#Assess Country Data

print('################################')
print('Changed :',CountryData.columns.values)
CountryData.rename(columns={'Country': 'Country_Name'}, inplace=True)
CountryData.rename(columns={'ISO-2-CODE': 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inplace=True)
CountryData.drop('ISO-3-Code', axis=1, inplace=True)
CountryData.drop('RowID', axis=1, inplace=True)

print('To :',CountryData.columns.values)
```

```
print('################################')

#Import Company Data

sFileName=Base + '/' + Company + '/' + sInputFileName2
print('################################')
print('Loading :',sFileName)
print('################################')

CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")

print('Loaded Company :',CompanyData.columns.values)
print('################################')

#Assess Company Data

print('################################')
print('Changed :',CompanyData.columns.values)
CompanyData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CompanyData.columns.values)
print('################################')

#Import Customer Data

sFileName=Base + '/' + Company + '/' + sInputFileName3
print('################################')
print('Loading :',sFileName)
print('################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
print('################################')

print('Loaded Customer :',CustomerRawData.columns.values)
print('################################')

CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('################################')
print('Remove Blank Country Code')
print('Reduce Rows from', CustomerRawData.shape[0],' to ',
CustomerData.shape[0])
print('################################')

print('################################')
print('Changed :',CustomerData.columns.values)
```

```
CustomerData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CustomerData.columns.values)
print('################################')

print('################################')
print('Merge Company and Country Data')
print('################################')
CompanyNetworkData=pd.merge(CompanyData,CountryData,how='inner',on='Country_Code
')

print('################################')
print('Change ',CompanyNetworkData.columns.values)
for i in CompanyNetworkData.columns.values:
    j='Company_'+i
    CompanyNetworkData.rename(columns={i: j}, inplace=True)
print('To ', CompanyNetworkData.columns.values)
print('################################')

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
CompanyNetworkData.to_csv(sFileName, index = False, encoding="latin-1")

print('################################')
print('### Done!! ####################')
print('################################')
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/pytho
y
################################
Working Base : C:/VKHCG  using Windows
################################
################################
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv
################################
Loaded Country: ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
################################
################################
Changed : ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
To : ['Country_Name' 'Country_Code']
################################
################################
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv
################################
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude']
################################
################################
Changed : ['Country' 'Place_Name' 'Latitude' 'Longitude']
To : ['Country_Code' 'Place_Name' 'Latitude' 'Longitude']
################################
################################
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv
################################
################################
Loaded Customer : ['version https://git-lfs.github.com/spec/v1']
################################
################################
Remove Blank Country Code
Reduce Rows from 2  to  2
################################
################################
Changed : ['version https://git-lfs.github.com/spec/v1']
To : ['version https://git-lfs.github.com/spec/v1']
################################
################################
Merge Company and Country Data
################################
```

```
################################
Change  ['Country_Code' 'Place_Name' 'Latitude' 'Longitude' 'Country_Name']
To  ['Company_Country_Code' 'Company_Place_Name' 'Company_Latitude'
 'Company_Longitude' 'Company_Country_Name']
################################
################################
Storing : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv
################################
################################
### Done!! ####################
################################
PS C:\Users\Dell\Desktop\Rahul>
```

Next, Access the the customers location using network router location.
####################Assess-Network-Routing-Customer#####################

```
      import sys
import os
import pandas as pd

pd.options.mode.chained_assignment = None

Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')

sInputFileName=Base+'/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-
Routing-Customer.csv'
sOutputFileName='Assess-Network-Routing-Customer.gml'
Company='01-Vermeulen'

sFileName=sInputFileName

print('################################')
print('Loading :',sFileName)
print('################################')

CustomerData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
print('Loaded Country:',CustomerData.columns.values)
print('################################')

print(CustomerData.head())

print('################################')
print('### Done!! ####################')
print('################################')
```

Output

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/D
py
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
################################
  Customer_Country_Code Customer_Place_Name  Customer_Latitude  Customer_Longitude Customer_Country_Name
0                    BW            Gaborone          -24.6464             25.9119              Botswana
1                    BW          Francistown          -21.1667             27.5167              Botswana
2                    BW                Maun          -19.9833             23.4167              Botswana
3                    BW           Molepolole          -24.4167             25.5333              Botswana
4                    NE               Niamey           13.5167              2.1167                 Niger
################################
### Done!! ####################
################################
PS C:\Users\Dell\Desktop\Rahul>
```

Assess-Network-Routing-Customer.csv

Assess-Network-Routing-Node.py

```
import sys
import os
import pandas as pd

pd.options.mode.chained_assignment = None

Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)

print('################################')

sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'

sOutputFileName='Assess-Network-Routing-Node.csv'
Company='01-Vermeulen'

sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')

IPData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded IP :', IPData.columns.values)
print('################################')

print('################################')
print('Changed :',IPData.columns.values)
IPData.drop('RowID', axis=1, inplace=True)
IPData.drop('ID', axis=1, inplace=True)
IPData.rename(columns={'Country': 'Country_Code'}, inplace=True)
IPData.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
IPData.rename(columns={'Post.Code': 'Post_Code'}, inplace=True)


IPData.rename(columns={'First.IP.Number': 'First_IP_Number'}, inplace=True)
IPData.rename(columns={'Last.IP.Number': 'Last_IP_Number'}, inplace=True)
print('To :',IPData.columns.values)
print('################################')

print('################################')
print('Change ',IPData.columns.values)
```

```
for i in IPData.columns.values:
    j='Node_'+i
    IPData.rename(columns={i: j}, inplace=True)
print('To ', IPData.columns.values)
print('################################')

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
IPData.to_csv(sFileName, index = False, encoding="latin-1")

print('################################')
print('### Done!! ####################')
print('################################')
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe
ode.py
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv
################################
Loaded IP : ['RowID' 'ID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude'
 'First.IP.Number' 'Last.IP.Number']
################################
################################
Changed : ['RowID' 'ID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude'
 'First.IP.Number' 'Last.IP.Number']
To : ['Country_Code' 'Place_Name' 'Post_Code' 'Latitude' 'Longitude'
 'First_IP_Number' 'Last_IP_Number']
################################
################################
Change  ['Country_Code' 'Place_Name' 'Post_Code' 'Latitude' 'Longitude'
 'First_IP_Number' 'Last_IP_Number']
To  ['Node_Country_Code' 'Node_Place_Name' 'Node_Post_Code' 'Node_Latitude'
 'Node_Longitude' 'Node_First_IP_Number' 'Node_Last_IP_Number']
################################
################################
Storing : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Node.csv
################################
################################
### Done!! ###################
################################
PS C:\Users\Dell\Desktop\Rahul>
```

C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Node.csv

Writeups:

Practical No: - 6

Practical 6:

Processing Data

A. Golden Nominal

A golden nominal record is a single person's record, with distinctive references for use by all systems. This gives the system a single view of the person. I use first name, other names, last name, and birth date as my golden nominal. The data we have in the assess directory requires a birth date to become a golden nominal. The proram will generate a golden nominal using our sample data set.

Open your Python editor and create a file called Process-People.py in the ..

```
import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
from datetime import datetime, timedelta
from pytz import timezone, all_timezones
from random import randint
import uuid
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
Company='04-Clark'
sInputFileName='02-Assess/01-EDS/02-Python/Assess_People.csv'
#################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn1 = sq.connect(sDatabaseName)
#################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
```

```
#####################################################################
### Import Female Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
print(sFileName)
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)

start_date = datetime(1900,1,1,0,0,0)
start_date_utc=start_date.replace(tzinfo=timezone('UTC'))

HoursBirth=100*365*24
RawData['BirthDateUTC']=RawData.apply(lambda row:
                (start_date_utc + timedelta(hours=randint(0, HoursBirth)))
                ,axis=1)
zonemax=len(all_timezones)-1
RawData['TimeZone']=RawData.apply(lambda row:
                (all_timezones[randint(0, zonemax)])
                ,axis=1)
RawData['BirthDateISO']=RawData.apply(lambda row:
                row["BirthDateUTC"].astimezone(timezone(row['TimeZone']))
                ,axis=1)
RawData['BirthDateKey']=RawData.apply(lambda row:
                row["BirthDateUTC"].strftime("%Y-%m-%d %H:%M:%S")
                ,axis=1)
RawData['BirthDate']=RawData.apply(lambda row:
                row["BirthDateISO"].strftime("%Y-%m-%d %H:%M:%S")
                ,axis=1)
RawData['PersonID']=RawData.apply(lambda row:
                str(uuid.uuid4())
                ,axis=1)

#####################################################################

Data=RawData.copy()
Data.drop('BirthDateUTC', axis=1,inplace=True)
Data.drop('BirthDateISO', axis=1,inplace=True)
indexed_data = Data.set_index(['PersonID'])

print('################################')
#####################################################################
```

```
print('################')
sTable='Process_Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_data.to_sql(sTable, conn1, if_exists="replace")
print('################')
#####################################################################
PersonHubRaw=Data[['PersonID','FirstName','SecondName','LastName','BirthDateKey'
]]
PersonHubRaw['PersonHubID']=RawData.apply(lambda row:
                str(uuid.uuid4())
                ,axis=1)
PersonHub=PersonHubRaw.drop_duplicates(subset=None, \
                                       keep='first',\
                                       inplace=False)
indexed_PersonHub = PersonHub.set_index(['PersonHubID'])
sTable = 'Hub-Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonHub.to_sql(sTable, conn1, if_exists="replace")
#####################################################################
PersonSatelliteGenderRaw=Data[['PersonID','FirstName','SecondName','LastName'\
                          ,'BirthDateKey','Gender']]
PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lambda row:
                str(uuid.uuid4())
                ,axis=1)
PersonSatelliteGender=PersonSatelliteGenderRaw.drop_duplicates(subset=None, \
                                                   keep='first', \
                                                   inplace=False)
indexed_PersonSatelliteGender =
PersonSatelliteGender.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Gender'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteGender.to_sql(sTable, conn1, if_exists="replace")
#####################################################################
PersonSatelliteBirthdayRaw=Data[['PersonID','FirstName','SecondName','LastName',
\
                            'BirthDateKey','TimeZone','BirthDate']]
PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lambda row:
                str(uuid.uuid4())
                ,axis=1)
PersonSatelliteBirthday=PersonSatelliteBirthdayRaw.drop_duplicates(subset=None,
\
                                                   keep='first',
\
```

```
                                                                inplace=False
)
indexed_PersonSatelliteBirthday =
PersonSatelliteBirthday.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Names'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteBirthday.to_sql(sTable, conn1, if_exists="replace")
#####################################################################
sFileDir=Base + '/' + Company + '/03-Process/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####################################################################
sOutputFileName = sTable + '.csv'
sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
#####################################################################
print('################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
print('################')
print('### Done!! ############################################')
#####################################################################
```

Output :

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/Desktop/Rahul/processingdata.py
##############################
Working Base : C:/VKHCG  using  win32
##############################
##############################
Loading : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
##############################
C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
##############################
###############
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process_Person
###############
c:\Users\Dell\Desktop\Rahul\processingdata.py:86: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  PersonHubRaw['PersonHubID']=RawData.apply(lambda row:
Storing : C:/VKHCG/88-DV/datavault.db  Table: Hub-Person
c:\Users\Dell\Desktop\Rahul\processingdata.py:99: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lambda row:
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Person-Gender
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Person-Names
##############################
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Satellite-Person-Names.csv
##############################
##############################
###############
Vacuum Databases
###############
### Done!! ############################################
```

It will apply golden nominal rules by assuming nobody born before January 1, 1900, droping to two ISO complex date time structures, as the code does not translate into SQLite's data types and saves your new golden nominal to a CSV file.

B. Vehicles

The international classification of vehicles is a complex process. There are standards, but these are not universally applied or similar between groups or countries.
Let's load the vehicle data for Hillman Ltd into the data vault, as we will need it later.
Create a new file named Process-Vehicle-Logistics.py in the Python editor in directory ..\VKHCG\03-Hillman\03-Process.

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid

pd.options.mode.chained_assignment = None
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='03-Hillman'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
################################################################

sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################

sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
################################################################

sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
```

```
#####################################################################

sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################

sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('############')
print('Loading :',sFileName)
VehicleRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#####################################################################

sTable='Process_Vehicles'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleRaw.to_sql(sTable, conn1, if_exists="replace")
#####################################################################

VehicleRawKey=VehicleRaw[['VehicleMake','VehicleModel']].copy()
VehicleKey=VehicleRawKey.drop_duplicates()
#####################################################################

VehicleKey['ObjectKey']=VehicleKey.apply(lambda row:
str('('+ str(row['VehicleMake']).strip().replace(' ', '-').replace('/', '-
').lower() +
')-(' + (str(row['VehicleModel']).strip().replace(' ', '-').replace(' ', '-
').lower())
+')')
,axis=1)

#####################################################################
VehicleKey['ObjectType']=VehicleKey.apply(lambda row:
'vehicle'
,axis=1)
#####################################################################

VehicleKey['ObjectUUID']=VehicleKey.apply(lambda row:
str(uuid.uuid4())
,axis=1)
#####################################################################
### Vehicle Hub
#####################################################################
#
```

```
VehicleSatellite=VehicleKey[['VehicelObjectType','VehicleObjectKey','VehicleObje
ctUUID','VehicleMake','VehicleModel']].copy()
VehicleSatellite.index.name='VehicleObjectSatelliteID'
sTable = 'VehicleSatellite-VehicleObject-VehicleMake-VehicleModel'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleSatellite.to_sql(sTable, conn1, if_exists="replace")


#####################################################################
### Vehicle Dimension
#####################################################################
sView='Dim-Object'
print('Storing :',sDatabaseName,' View:',sView)
sSQL="CREATE VIEW IF NOT EXISTS [" + sView + "] AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   H.ObjectType,"
sSQL=sSQL+ "   H.ObjectKey AS VehicleKey,"
sSQL=sSQL+ "   TRIM(S.Make) AS VehicleMake,"
sSQL=sSQL+ "   TRIM(S.Model) AS VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   [Hub-Object-Vehicle] AS H"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ "   [Satellite-Object-Make-Model] AS S"
sSQL=sSQL+ " ON"
sSQL=sSQL+ "     H.ObjectType=S.ObjectType"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " H.ObjectUUID=S.ObjectUUID;"
sql.execute(sSQL,conn1)

print('################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " VehicleMake,"
sSQL=sSQL+ " VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " [" + sView + "]"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " VehicleMake"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " VehicleMake;"
DimObjectData=pd.read_sql_query(sSQL, conn1)

DimObjectData.index.name='ObjectDimID'
```

```
DimObjectData.sort_values(['VehicleMake','VehicleModel'],inplace=True,
ascending=True)
print('#################')
print(DimObjectData)
#########################################################################
print('#################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)

print('#################')
#########################################################################
conn1.close()
conn2.close()
#########################################################################
#print('### Done!! ############################################')
#########################################################################
```

Output :

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Pyth
ics.py
####################################
Working Base : C:/VKHCG  using  win32
####################################
############
Loading : C:/VKHCG/03-Hillman/00-RawData/VehicleData.csv
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process_Vehicles
Storing : C:/VKHCG/88-DV/datavault.db  Table: Hub-Object-Vehicle
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Object-Make-Model
Storing : C:/VKHCG/88-DV/datavault.db  View: Dim-Object
#################
Loading : C:/VKHCG/88-DV/datavault.db  Table: Dim-Object
#################
                VehicleMake                          VehicleModel
ObjectDimID
2213             AM General                 DJ Po Vehicle 2WD
2212             AM General                   FJ8c Post Office
129              AM General               Post Office DJ5 2WD
131              AM General               Post Office DJ8 2WD
2869       ASC Incorporated                               GNX
...                     ...                               ...
1996                 smart                fortwo convertible
1997                 smart                      fortwo coupe
2622                 smart    fortwo electric drive cabriolet
2833                 smart  fortwo electric drive convertible
2623                 smart        fortwo electric drive coupe

[3885 rows x 2 columns]
#################
Vacuum Databases
#################
PS C:\Users\Dell\Desktop\Rahul>
```

C. Human-Environment Interaction

In the Python editor, open a new file named Process_Location.py in directory ..\VKHCG\01-Vermeulen\03-Process.

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='01-Vermeulen'
InputAssessGraphName='Assess_All_Animals.gml'
EDSAssessDir='02-Assess/01-EDS'
InputAssessDir=EDSAssessDir + '/02-Python'
################################################################
sFileAssessDir=Base + '/' + Company + '/' + InputAssessDir
if not os.path.exists(sFileAssessDir):
    os.makedirs(sFileAssessDir)
################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
################################################################
```

```
t=0
tMax=360*180
#####################################################################
for Longitude in range(-180,180,10):
    for Latitude in range(-90,90,10):
        t+=1
        IDNumber=str(uuid.uuid4())
        LocationName='L'+format(round(Longitude,3)*1000, '+07d') +\
                                '-'+format(round(Latitude,3)*1000, '+07d')
        print('Create:',t,' of ',tMax,':',LocationName)
        LocationLine=[('ObjectBaseKey', ['GPS']),
                ('IDNumber', [IDNumber]),
                ('LocationNumber', [str(t)]),
                ('LocationName', [LocationName]),
                ('Longitude', [Longitude]),
                ('Latitude', [Latitude])]
        if t==1:
            LocationFrame =
pd.DataFrame(columns=['ObjectBaseKey','IDNumber','LocationNumber','LocationName'
,'Logitude','Latitude'])
        else:
            LocationRow =
pd.DataFrame(columns=['ObjectBaseKey','IDNumber','LocationNumber','LocationName'
,'Logitude','Latitude'])
            LocationFrame = LocationFrame.append(LocationRow)

#####################################################################
LocationHubIndex=LocationFrame.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Process-Location'
print('Storing :',sDatabaseName,' Table:',sTable)
LocationHubIndex.to_sql(sTable, conn1, if_exists="replace")
#####################################################################
sTable = 'Hub-Location'
print('Storing :',sDatabaseName,' Table:',sTable)
LocationHubIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('################')
#####################################################################
```

```
print('### Done!! ############################################')
########################################################################
```

Output :

```
Create: 644  of  64800 : L+170000-+040000
c:\Users\Dell\Desktop\Rahul\HumanEnvironment.py:62: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a futur
e version. Use pandas.concat instead.
  LocationFrame = LocationFrame.append(LocationRow)
Create: 645  of  64800 : L+170000-+050000
c:\Users\Dell\Desktop\Rahul\HumanEnvironment.py:62: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a futur
e version. Use pandas.concat instead.
  LocationFrame = LocationFrame.append(LocationRow)
Create: 646  of  64800 : L+170000-+060000
c:\Users\Dell\Desktop\Rahul\HumanEnvironment.py:62: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a futur
e version. Use pandas.concat instead.
  LocationFrame = LocationFrame.append(LocationRow)
Create: 647  of  64800 : L+170000-+070000
c:\Users\Dell\Desktop\Rahul\HumanEnvironment.py:62: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a futur
e version. Use pandas.concat instead.
  LocationFrame = LocationFrame.append(LocationRow)
Create: 648  of  64800 : L+170000-+080000
c:\Users\Dell\Desktop\Rahul\HumanEnvironment.py:62: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a futur
e version. Use pandas.concat instead.
  LocationFrame = LocationFrame.append(LocationRow)
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process-Location
Storing : C:/VKHCG/88-DV/datavault.db  Table: Hub-Location
###############
Vacuum Databases
###############
### Done!! ############################################
PS C:\Users\Dell\Desktop\Rahul>
```

D. Forecasting

Forecasting is the ability to project a possible future, by looking at historical data. The datavault enables these types of investigations, owing to the complete history it collects as itprocesses the source's systems data. A data scientist supply answers to such questions as the following:

- What should we buy?
- What should we sell?
- Where will our next business come from?

People want to know what you calculate to determine what is about to happen.

Open a new file in your Python editor and save it as Process-Shares-Data.py in directory C: \VKHCG\04-Clark\03-Process. I will guide you through this process.

You will require a library called quandl **type pip install quandl in cmd**

```
################################################################
import sys
import os
import sqlite3 as sq
import quandl
import pandas as pd
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='04-Clark'
sInputFileName='00-RawData/VKHCG_Shares.csv'
sOutputFileName='Shares.csv'
################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sFileDir1=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir1):
    os.makedirs(sFileDir1)
################################################################
sFileDir2=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir2):
    os.makedirs(sFileDir2)
```

```
#####################################################################
sFileDir3=Base + '/' + Company + '/03-Process/01-EDS/02-Python'
if not os.path.exists(sFileDir3):
    os.makedirs(sFileDir3)
#####################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
#####################################################################
### Import Share Names Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Rows    :',RawData.shape[0])
print('Columns:',RawData.shape[1])
print('################')
#####################################################################
sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
#####################################################################
sFileName=sFileDir2 + '/Assess_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
#####################################################################
sFileName=sFileDir3 + '/Process_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
#####################################################################
### Import Shares Data Details
#####################################################################
nShares=RawData.shape[0]
```

```
#nShares=6
for sShare in range(nShares):
    sShareName=str(RawData['Shares'][sShare])
    ShareData = quandl.get(sShareName)
    UnitsOwn=RawData['Units'][sShare]
    ShareData['UnitsOwn']=ShareData.apply(lambda row:(UnitsOwn),axis=1)
    ShareData['ShareCode']=ShareData.apply(lambda row:(sShareName),axis=1)
    print('################')
    print('Share  :',sShareName)
    print('Rows   :',ShareData.shape[0])
    print('Columns:',ShareData.shape[1])
    print('################')
    ######################################################################
    print('################')
    sTable=str(RawData['sTable'][sShare])
    print('Storing :',sDatabaseName,' Table:',sTable)
    ShareData.to_sql(sTable, conn, if_exists="replace")
    print('################')
    ######################################################################
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
    print('################################')
    ######################################################################
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir2 + '/Assess_' + sOutputFileName
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
    print('################################')
    ######################################################################
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir3 + '/Process_' + sOutputFileName
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
    print('################################')
######################################################################
######################################################################
```

```
print('### Done!! ############################################')
##########################################################################
```

Output:

```
PS C:\Users\Dell\OneDrive\Documents\ric> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe
sting.py
##############################
Working Base : C:/VKHCG  using  win32
##############################
##############################
Loading : C:/VKHCG/04-Clark/00-RawData/VKHCG_Shares.csv
##############################
Rows    : 10
Columns: 3
###############
##############################
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_Shares.csv
##############################
##############################
##############################
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_Shares.csv
##############################
##############################
##############################
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_Shares.csv
##############################
##############################
```

Writeups:

Practical No: - 7

Practical 7:
Transforming Data

Transform Superstep

C: \VKHCG\01-Vermeulen\04-Transform.

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid

pd.options.mode.chained_assignment = None
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen'
conn1 = sq.connect(sDatabaseName)
################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
    os.makedirs(sDataVaultDir)
################################################################
```

```
sDatabaseName=sDataVaultDir + '/datavault'
conn2 = sq.connect(sDatabaseName)
#################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
#################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse'
conn3 = sq.connect(sDatabaseName)
#################################################################
print('\n#################################')
print('Time Category')
print('UTC Time')
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
print(BirthDateZoneUTCStr)
print('#################################')

print('Birth Date in Reykjavik :')
BirthZone = 'Atlantic/Reykjavik'
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
print(BirthDateStr)
print('#################################')
#################################################################
IDZoneNumber=str(uuid.uuid4())
sDateTimeKey=BirthDateZoneStr.replace(' ','-').replace(':','-')
TimeLine=[('ZoneBaseKey', ['UTC']),
              ('IDNumber', [IDZoneNumber]),
              ('DateTimeKey', [sDateTimeKey]),
              ('UTCDateTimeValue', [BirthDateZoneUTC]),
              ('Zone', [BirthZone]),
              ('DateTimeValue', [BirthDateStr])]
TimeFrame = pd.DataFrame(columns =
['ZoneBaseKey','IDNumber','DateTimeKey','UTCDateTimeValue','Zone','DateTimeValue
'])
#################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#################################################################
sTable = 'Hub-Time-Gunnarsson'
```

```
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Time-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#################################################################
TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']]
TimeSatelliteIndex=TimeSatellite.set_index(['IDNumber'],inplace=False)
#################################################################
BirthZoneFix=BirthZone.replace(' ','-').replace('/','-')
sTable = 'Satellite-Time-' + BirthZoneFix + '-Gunnarsson'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeSatelliteIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Time-' + BirthZoneFix + '-Gunnarsson'
TimeSatelliteIndex.to_sql(sTable, conn3, if_exists="replace")
#################################################################
print('\n#################################')
print('Person Category')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
print('Name:',FirstName,LastName)
print('Birth Date:',BirthDateLocal)
print('Birth Zone:',BirthZone)
print('UTC Birth Date:',BirthDateZoneStr)
print('#################################')
#################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('IDNumber', [IDPersonNumber]),
            ('FirstName', [FirstName]),
            ('LastName', [LastName]),
            ('Zone', ['UTC']),
            ('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame(columns =
['IDNumber','FirstName','LastName','Zone','DateTimeValue'])
#################################################################
TimeHub=PersonFrame
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#################################################################
sTable = 'Hub-Person-Gunnarsson'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
```

```
print('\n###################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Person-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
```

Output : Guðmundur Gunnarsson was born on December 20, 1960, at 9:15 in Landspítali,Hringbraut 101, 101 Reykjavík, Iceland.

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/L
cs.py
################################
Working Base : C:/VKHCG  using  win32
################################
###########
Loading : C:/VKHCG/03-Hillman/00-RawData/VehicleData.csv
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/L
orn.py
################################
Working Base : C:/VKHCG  using  win32
################################

################################
Time Category
UTC Time
1960-12-20 10:15:00 (UTC) (+0000)
################################
Birth Date in Reykjavik :
1960-12-20 10:15:00 (GMT) (+0000)
################################

################################
Storing : C:/VKHCG/99-DW/datawarehouse
 Table: Hub-Time-Gunnarsson

################################

################################
Storing : C:/VKHCG/99-DW/datawarehouse
 Table: Satellite-Time-Atlantic-Reykjavik-Gunnarsson

################################

################################
Person Category
Name: Guðmundur Gunnarsson
```

```
Birth Date: 1960-12-20 10:15:00
Birth Zone: Atlantic/Reykjavik
UTC Birth Date: 1960-12-20 10:15:00
################################

################################
Storing : C:/VKHCG/99-DW/datawarehouse
 Table: Hub-Person-Gunnarsson

################################
PS C:\Users\Dell\Desktop\Rahul>
```

You must build three items: **dimension Person**, **dimension Time**, and **factPersonBornAtTime**. Open your Python editor and create a file named Transform-Gunnarsson-Sun-Model.py in directory C:\VKHCG\01 Vermeulen\04-Transform.

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='01-Vermeulen'
################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
```

```
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
#################################################################
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir):
    os.makedirs(sDataWarehousetDir)
#################################################################
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
conn2 = sq.connect(sDatabaseName)
#################################################################
print('\n#################################')
print('Time Dimension')
BirthZone = 'Atlantic/Reykjavik'
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
#################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [IDTimeNumber]),
          ('UTCDate', [BirthDateZoneStr]),
          ('LocalTime', [BirthDateLocal]),
          ('TimeZone', [BirthZone])]
TimeFrame = pd.DataFrame(columns = ['TimeID' , 'UTCDate' , 'LocalTime' ,
'TimeZone'])
#################################################################
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
#################################################################
sTable = 'Dim-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('\n#################################')
print('Dimension Person')
print('\n#################################')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
```

```
#####################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [IDPersonNumber]),
            ('FirstName', [FirstName]),
            ('LastName', [LastName]),
            ('Zone', ['UTC']),
            ('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame(columns = ['PersonID' , 'FirstName' , 'LastName' ,
'Zone' , 'DateTimeValue'])
#####################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####################################################################
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('\n#################################')
print('Fact - Person - time')
print('\n#################################')
IDFactNumber=str(uuid.uuid4())
PersonTimeLine=[('IDNumber', [IDFactNumber]),
                ('IDPersonNumber', [IDPersonNumber]),
                ('IDTimeNumber', [IDTimeNumber])]
PersonTimeFrame = pd.DataFrame(columns = ['IDNumber' , 'IDPersonNumber' ,
'IDTimeNumber'])
#####################################################################
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Fact-Person-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/Downloads/Transform-Gunnarsson-Sun-
Model.py
##############################
Working Base : C:/VKHCG  using  win32
##############################

##############################
Time Dimension

##############################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Dim-Time

##############################
```

Writeups:

Practical No: - 8

Practical 8:
Organizing Data

C:\VKHCG\01-Vermeulen\05-Organise\ OrganizeHorizontal.py

A. Horizontal Style.

```
################################################################
########
# -*- coding: utf-8 -*-
################################################################
########
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
########
if sys.platform == 'linux':
 Base=os.path.expanduser('~') + '/VKHCG'
else:
 Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
########
################################################################
########
Company='01-Vermeulen'
################################################################
########
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
 os.makedirs(sDataWarehouseDir)
################################################################
########
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
################################################################
########
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
```

```
############################################################
########
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn2)
############################################################
########
print('################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT PersonID,\
 Height,\
 Weight,\
 bmi,\
 Indicator\
 FROM [Dim-BMI]\
 WHERE \
 Height > 1.5 \
 and Indicator = 1\
 ORDER BY \
 Height,\
 Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn2)
############################################################
########
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
############################################################
########
sTable = 'Dim-BMI-Horizontal'
print('\n################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
############################################################
########
print('################################')
sTable = 'Dim-BMI-Horizontal'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT * FROM [Dim-BMI];"
```

```
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#############################################################
########
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('################################')
#############################################################
########
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe
################################
Working Base : C:/VKHCG  using  win32
################################
###############
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################

################################
Storing : C:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI-Horizontal

################################
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Horizontal
################################
################################
Full Data Set (Rows): 194
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 5
################################
PS C:\Users\Dell\Desktop\Rahul>
```

B. Vertical Style

C:\VKHCG\01-Vermeulen\05-Organise
Organize-Vertical.py

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
################################################################
Company='01-Vermeulen'
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
################################################################
print('################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn2)
################################################################
print('################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
```

```
print('################################')
sSQL="SELECT \
       Height,\
       Weight,\
       Indicator\
  FROM [Dim-BMI];"
PersonFrame1=pd.read_sql_query(sSQL, conn2)
#####################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####################################################################
sTable = 'Dim-BMI-Vertical'
print('\n################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('###############')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#####################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('################################')
#####################################################################
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe
############################
Working Base : C:/VKHCG  using  win32
############################
############################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
############################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
############################

############################
Storing : C:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI-Vertical

############################
##############
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Vertical
############################
Full Data Set (Rows): 194
Full Data Set (Columns): 5
############################
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 3
############################
PS C:\Users\Dell\Desktop\Rahul>
```

C. Island Style

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Island.py

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
################################################################
Company='01-Vermeulen'
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn2)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
```

```
sSQL="SELECT \
       Height,\
       Weight,\
       Indicator,\
       CASE Indicator\
       WHEN 1 THEN 'Pip'\
       WHEN 2 THEN 'Norman'\
       WHEN 3 THEN 'Grant'\
       ELSE 'Sam'\
       END AS Name\
  FROM [Dim-BMI]\
  WHERE Indicator > 2\
  ORDER BY  \
       Height,\
       Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn2)
#################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#################################################################
sTable = 'Dim-BMI-Secure'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('################################')
sTable = 'Dim-BMI-Secure'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('Only Sam Data')
print(PersonFrame2.head())
print('################################')
```

```
#########################################################################
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe
##############################
Working Base : C:/VKHCG  using  win32
##############################
###############
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
###############
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

##############################
Storing : C:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI-Secure

##############################
##############################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Secure
##############################
##############################
Full Data Set (Rows): 194
Full Data Set (Columns): 5
##############################
Horizontal Data Set (Rows): 0
Horizontal Data Set (Columns): 4
Only Sam Data
Empty DataFrame
Columns: [Indicator, Height, Weight, Name]
Index: []
##############################
PS C:\Users\Dell\Desktop\Rahul>
```

D. Secure Vault Style
C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Secure-Vault.py

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
################################################################
Company='01-Vermeulen'
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn2)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="SELECT \
```

```
      Height,\
      Weight,\
      Indicator,\
      CASE Indicator\
      WHEN 1 THEN 'Pip'\
      WHEN 2 THEN 'Norman'\
      WHEN 3 THEN 'Grant'\
      ELSE 'Sam'\
      END AS Name\
  FROM [Dim-BMI]\
  WHERE Indicator > 2\
  ORDER BY  \
      Height,\
      Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn2)
#####################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####################################################################
sTable = 'Dim-BMI-Secure'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('################################')
sTable = 'Dim-BMI-Secure'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#####################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('Only Sam Data')
print(PersonFrame2.head())
print('################################')
#####################################################################
```

Output:

```
Working Base : C:/VKHCG  using  win32
##################################
#################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
#################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

##################################
Storing : C:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI-Secure

##################################
##################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Secure
##################################
##################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
##################################
Horizontal Data Set (Rows): 692
Horizontal Data Set (Columns): 4
Only Sam Data
   Indicator  Height  Weight Name
0          4     1.0      35  Sam
1          4     1.0      40  Sam
2          4     1.0      45  Sam
3          4     1.0      50  Sam
4          4     1.0      55  Sam
##################################
```

E. Association Rule Mining

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Association-Rule.py

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='01-Vermeulen'
InputFileName='Online-Retail-Billboard.xlsx'
EDSAssessDir='02-Assess/01-EDS'
InputAssessDir=EDSAssessDir + '/02-Python'
################################################################
sFileAssessDir=Base + '/' + Company + '/' + InputAssessDir
if not os.path.exists(sFileAssessDir):
    os.makedirs(sFileAssessDir)
################################################################
sFileName=Base+'/'+ Company + '/00-RawData/' + InputFileName
################################################################
df = pd.read_excel(sFileName)
print(df.shape)
################################################################
df['Description'] = df['Description'].str.strip()
df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]

basket = (df[df['Country'] =="France"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
```

```
          .set_index('InvoiceNo'))
##########################################################################
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1
##########################################################################
basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)

frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
print(rules.head())

rules[ (rules['lift'] >= 6) &
       (rules['confidence'] >= 0.8) ]
##########################################################################
sProduct1='ALARM CLOCK BAKELIKE GREEN'
print(sProduct1)
print(basket[sProduct1].sum())
sProduct2='ALARM CLOCK BAKELIKE RED'
print(sProduct2)
print(basket[sProduct2].sum())
##########################################################################
basket2 = (df[df['Country'] =="Germany"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))

basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)

print(rules2[ (rules2['lift'] >= 4) &
        (rules2['confidence'] >= 0.5)])
##########################################################################
print('### Done!! ############################################')
##########################################################################
```

Output:

```
PS C:\Users\Dell\Desktop\Rahul> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Dell/OneDrive/Documents/ric/mining.py
################################
Working Base : C:/VKHCG  using  win32
################################
(541909, 8)
C:\Users\Dell\AppData\Local\Programs\Python\Python310\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:111: DeprecationWarning: DataFrames w
ith non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool
type
  warnings.warn(
                  antecedents                  consequents  antecedent support  consequent support  ...  confidence      lift  leverage  convict
ion
0   (ALARM CLOCK BAKELIKE PINK)  (ALARM CLOCK BAKELIKE GREEN)            0.102041            0.096939  ...    0.725000  7.478947  0.064088    3.283
859
1  (ALARM CLOCK BAKELIKE GREEN)   (ALARM CLOCK BAKELIKE PINK)            0.096939            0.102041  ...    0.763158  7.478947  0.064088    3.791
383
2  (ALARM CLOCK BAKELIKE GREEN)    (ALARM CLOCK BAKELIKE RED)            0.096939            0.094388  ...    0.815789  8.642959  0.069932    4.916
181
3    (ALARM CLOCK BAKELIKE RED)  (ALARM CLOCK BAKELIKE GREEN)            0.094388            0.096939  ...    0.837838  8.642959  0.069932    5.568
878
4   (ALARM CLOCK BAKELIKE PINK)    (ALARM CLOCK BAKELIKE RED)            0.102041            0.094388  ...    0.725000  7.681081  0.064348    3.293
135

[5 rows x 9 columns]
ALARM CLOCK BAKELIKE GREEN
340.0
ALARM CLOCK BAKELIKE RED
316.0
C:\Users\Dell\AppData\Local\Programs\Python\Python310\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:111: DeprecationWarning: DataFrames w
ith non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool
type
  warnings.warn(
                     antecedents                       consequents  antecedent support  ...      lift  leverage  conviction
1   (PLASTERS IN TIN CIRCUS PARADE)  (PLASTERS IN TIN WOODLAND ANIMALS)            0.115974  ...  4.242887  0.051846    2.076984
7        (PLASTERS IN TIN SPACEBOY)  (PLASTERS IN TIN WOODLAND ANIMALS)            0.107221  ...  4.145125  0.046488    2.011670
11    (RED RETROSPOT CHARLOTTE BAG)            (WOODLAND CHARLOTTE BAG)            0.070022  ...  6.648168  0.050194    5.587746

[3 rows x 9 columns]
### Done!! ############################################
PS C:\Users\Dell\Desktop\Rahul>
```

Writeups:

Practical No: - 9

Practical 9
Generating Data

Report Superstep

The Report superstep is the step in the ecosystem that enhances the data science findings with the art of storytelling and data visualization. You can perform the best data science, but if you cannot execute a respectable and trustworthy Report step by turning your data science into actionable business insights, you have achieved no advantage for your business.

Vermeulen PLC

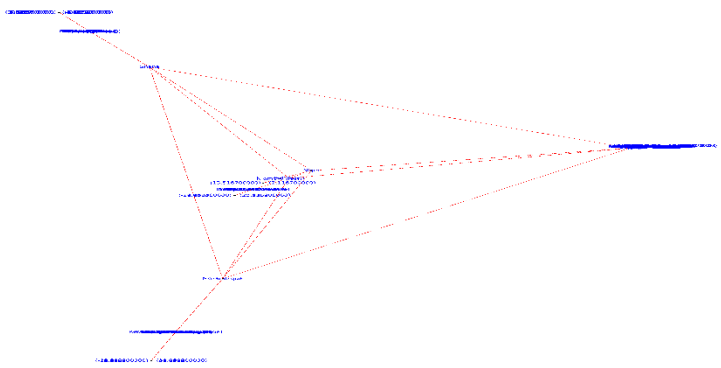
Vermeulen requires a map of all their customers' data links. Can you provide a report to deliver this? I will guide you through an example that delivers this requirement.
C:\VKHCG\01-Vermeulen\06-Report\Raport-Network-RoutingCustomer.py

```
################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
################################################################
pd.options.mode.chained_assignment = None
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv'
################################################################
sOutputFileName1='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml'
sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png'
Company='01-Vermeulen'
################################################################
################################################################
### Import Country Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
```

```
CustomerDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
CustomerData=CustomerDataRaw.head(100)
print('Loaded Country:',CustomerData.columns.values)
print('################################')
#####################################################################
print(CustomerData.head())
print(CustomerData.shape)
#####################################################################
G=nx.Graph()




print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
#####################################################################
sFileName=Base + '/' + Company + '/' + sOutputFileName1
print('################################')
print('Storing :',sFileName)
print('################################')
nx.write_gml(G, sFileName)
#####################################################################
sFileName=Base + '/' + Company + '/' + sOutputFileName2
print('################################')
print('Storing Graph Image:',sFileName)
print('################################')

plt.figure(figsize=(25, 25))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
#####################################################################
print('################################')
print('### Done!! ####################')
print('################################')
#####################################################################
```

Output:

```
PS C:\Users\Dell\OneDrive\Documents\ric> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe c
twork.py
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
################################
  Customer_Country_Code Customer_Place_Name  Customer_Latitude  Customer_Longitude Customer_Country_Name
0                    BW           Gaborone          -24.6464             25.9119              Botswana
1                    BW        Francistown          -21.1667             27.5167              Botswana
2                    BW               Maun          -19.9833             23.4167              Botswana
3                    BW         Molepolole          -24.4167             25.5333              Botswana
4                    NE             Niamey           13.5167              2.1167                 Niger
(100, 5)
Nodes: 0
Edges: 0
################################
Storing : C:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml
################################
################################
Storing Graph Image: C:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png
################################
################################
### Done!! ####################
################################
PS C:\Users\Dell\OneDrive\Documents\ric>
```



B. Krennwallner AG

The Krennwallner marketing department wants to deploy the locations of the billboards onto the company web server. Can you prepare three versions of the locations' web pages?

- Locations clustered into bubbles when you zoom out
- Locations as pins
- Locations as heat map

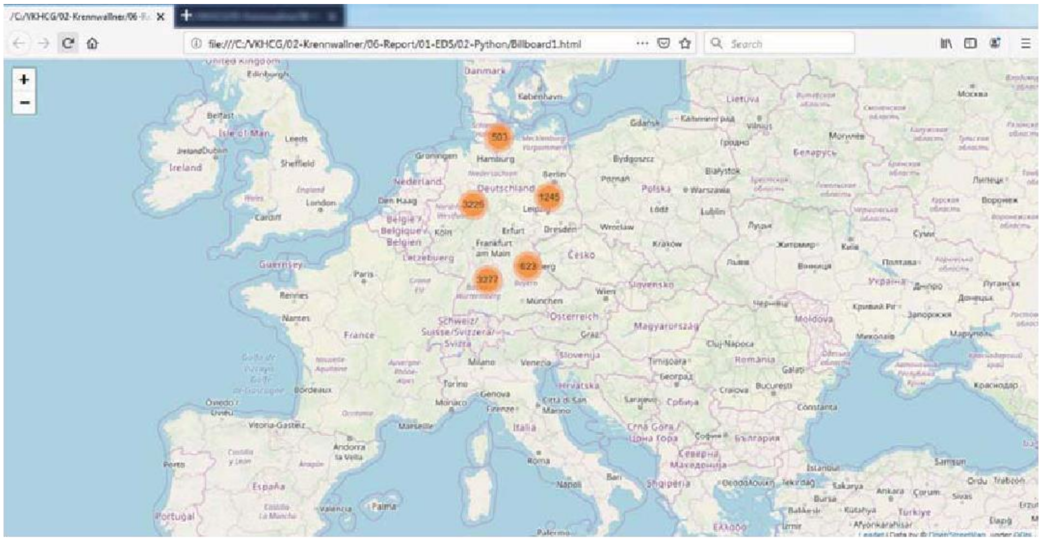
Picking Content for Billboards

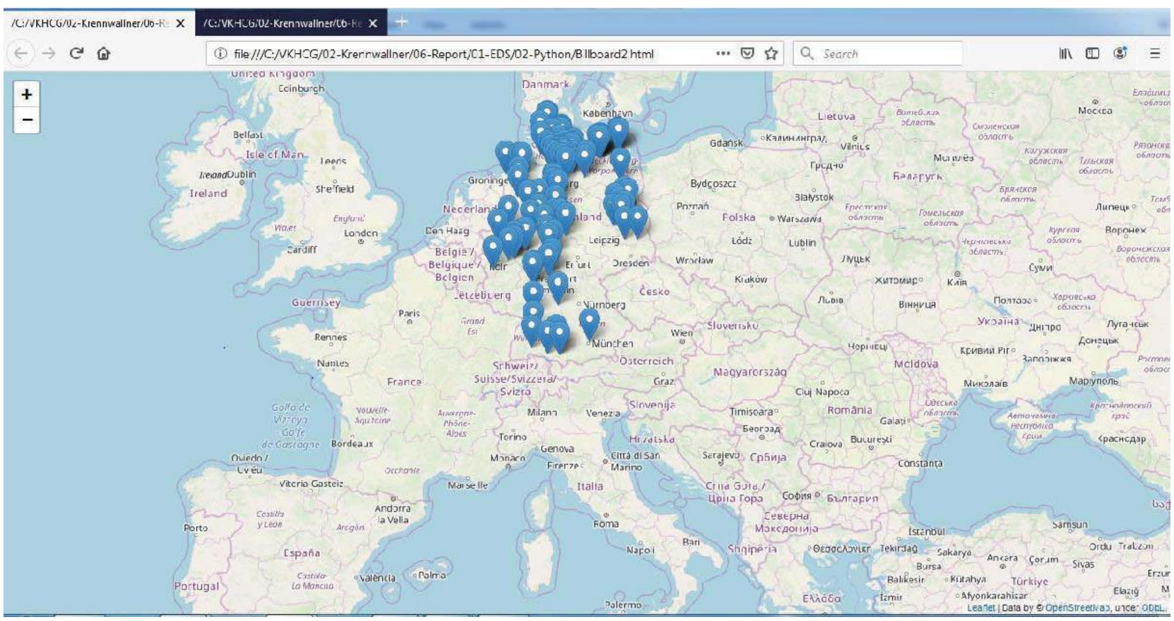
C:\VKHCG\02-Krennwallner\06-Report\Report_Billboard.py

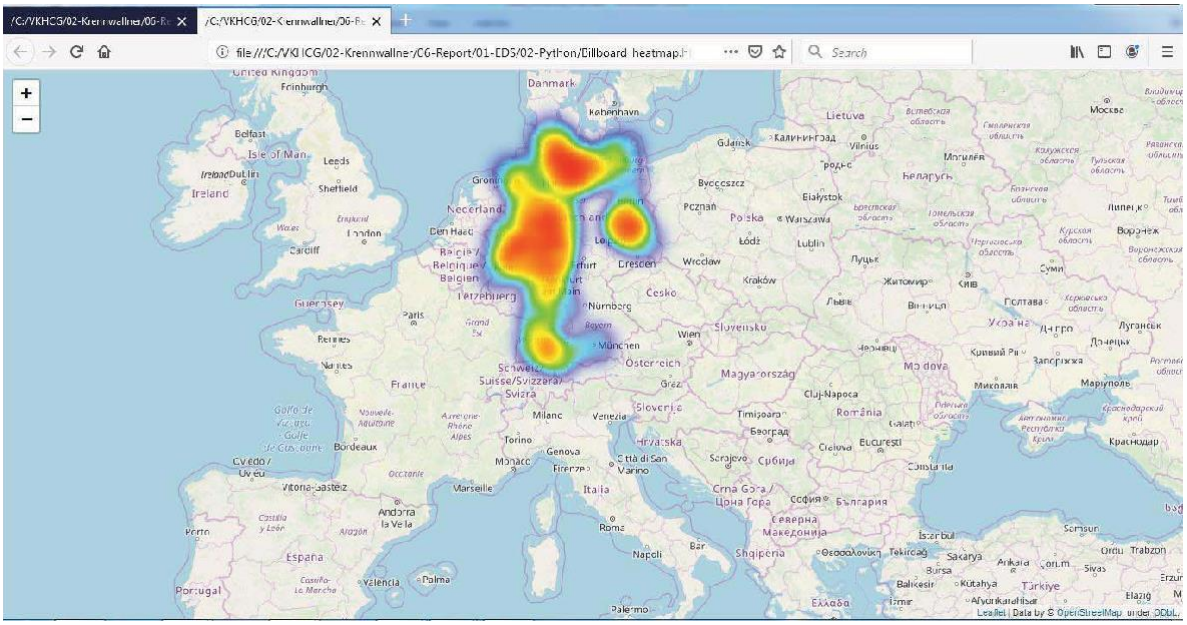
```
import sys import os import pandas as pd
from folium.plugins import FastMarkerCluster,
HeatMap from folium import Marker, Map import
webbrowser
################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
sFileName=Base+'/02-Krennwallner/01-Retrieve/01-
EDS/02Python/Retrieve_DE_Billboard_Locations.csv' df =
pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
df.fillna(value=0, inplace=True) print(df.shape)
################################################################
t=0 for i in range(df.shape[0]):
  try:
    sLongitude=df["Longitude"][i]
    sLongitude=float(sLongitude)
  except Exception:
    sLongitude=float(0.0) try:
    sLatitude=df["Latitude"][i]
    sLatitude=float(sLatitude)
  except Exception:
    sLatitude=float(0.0)
  try:
    sDescription=df["Place_Name"][i] + ' (' +
  df["Country"][i]+')' except Exception:
    sDescription='VKHCG'
  if sLongitude != 0.0 and sLatitude != 0.0:
    DataClusterList=list([sLatitude, sLongitude])
    DataPointList=list([sLatitude, sLongitude, sDescription])
    t+=1 if t==1:
      DataCluster=[DataClusterList]
```

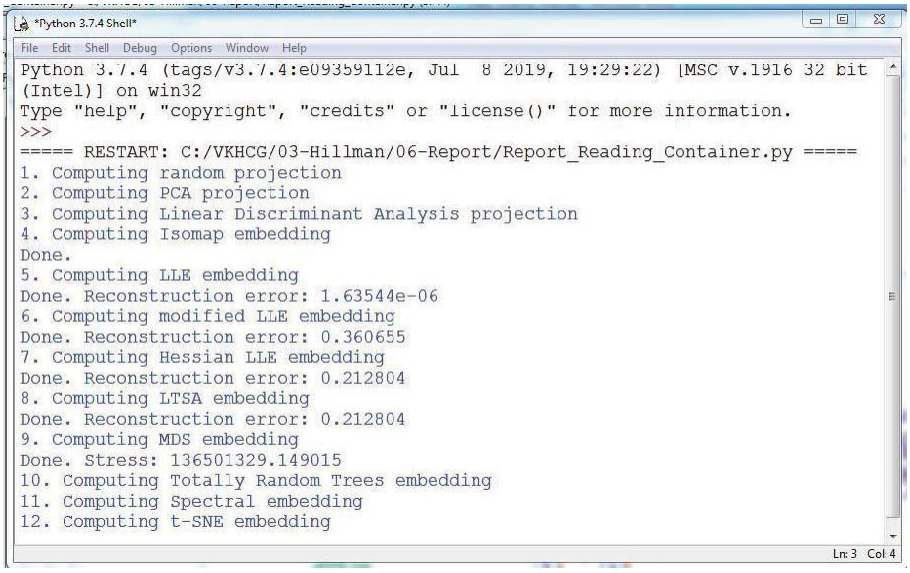
```
        DataPoint=[DataPointList] else:
        DataCluster.append(DataClusterList)
        DataPoint.append(DataPointList)
data=DataCluster
pins=pd.DataFrame(DataPoint)
pins.columns = [ 'Latitude','Longitude','Description']
############################################################
###### stops_map1 = Map(location=[48.1459806, 11.4985484],
zoom_start=5) marker_cluster =
FastMarkerCluster(data).add_to(stops_map1)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-
 Python/Billboard1.html'         stops_map1.save(sFileNameHtml)
webbrowser.open('file://'    +    os.path.realpath(sFileNameHtml))
#########################################################
############ stops_map2 = Map(location=[48.1459806,
  11.4985484], zoom_start=5)     for name,
row in pins.iloc[:100].iterrows():
   Marker([row["Latitude"],row["Longitude"]],
popup=row["Description"]).add_to(stops_map2) sFileNameHtml=Base+'/02-
Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html'
  stops_map2.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
#########################################################
############ stops_heatmap = Map(location=[48.1459806,
11.4985484],                                                zoom_start=5)
stops_heatmap.add_child(HeatMap([[row["Latitude"], row["Longitude"]]
for name, row in pins.iloc[:100].iterrows()])) sFileNameHtml=Base+'/02-
Krennwallner/06-Report/01-EDS/02-
Python/Billboard_heatmap.html' stops_heatmap.save(sFileNameHtml)
webbrowser.open('file://'        +        os.path.realpath(sFileNameHtml))
#########################################################
############ print('### Done!!
#############################################')
#########################################################
############
```

Output:


file:///C:/VKHCG/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard1.html
Search
503
3226
1245
623
3277


file:///C:/VKHCG/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html
Search


Search
Ireland
Belgien
France
España
Portugal
Italia
Ελλάδα
Leaflet | Data by © OpenStreetMap, under ODbL.


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/VKHCG/03-Hillman/06-Report/Report_Reading_Container.py =====
1. Computing random projection
2. Computing PCA projection
3. Computing Linear Discriminant Analysis projection
4. Computing Isomap embedding
Done.
5. Computing LLE embedding
Done. Reconstruction error: 1.63544e-06
6. Computing modified LLE embedding
Done. Reconstruction error: 0.360655
7. Computing Hessian LLE embedding
Done. Reconstruction error: 0.212804
8. Computing LTSA embedding
Done. Reconstruction error: 0.212804
9. Computing MDS embedding
Done. Stress: 136501329.149015
10. Computing Totally Random Trees embedding
11. Computing Spectral embedding
12. Computing t-SNE embedding
Ln: 3 Col: 4

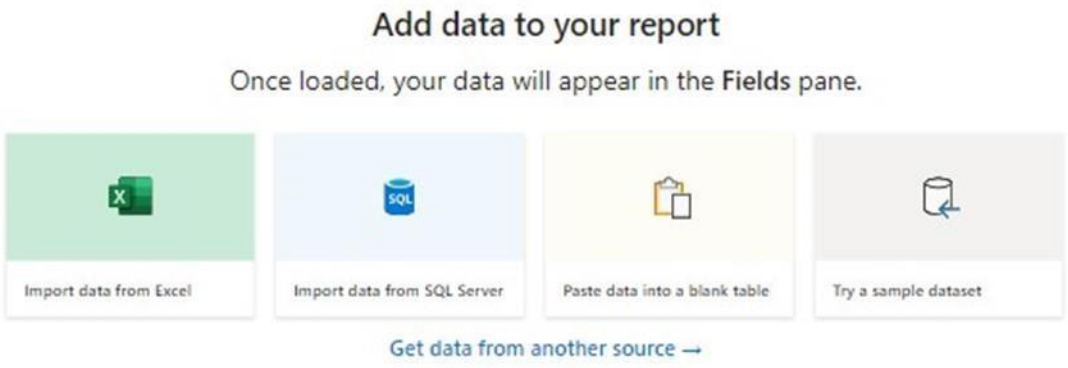
Writeups:

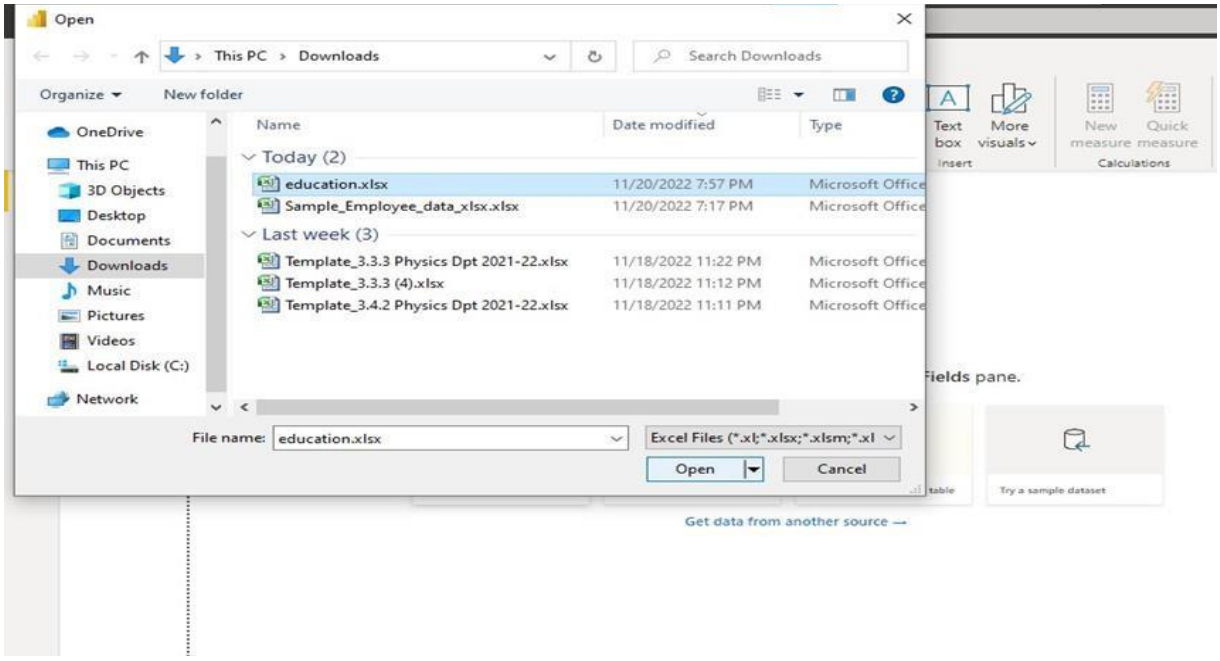
Practical No: - 10

Practical No: - 10

Step1:

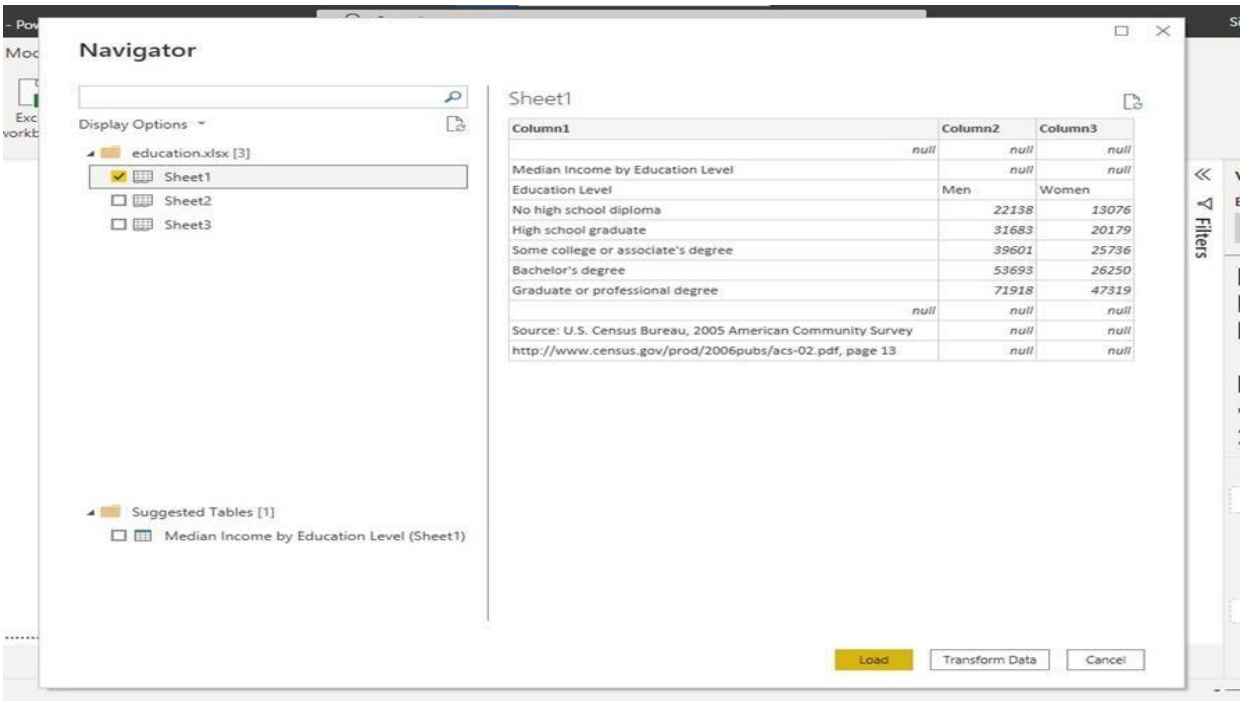
Open the power B.I and import the data **education.xlsx**





Step 2:

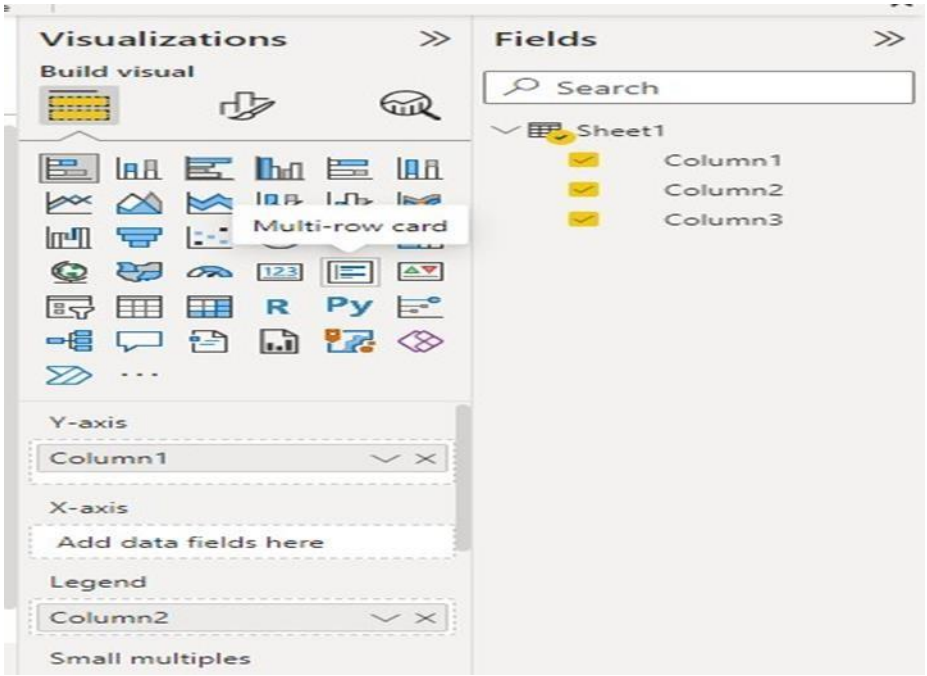
Now click on sheet 1 and Load the file.



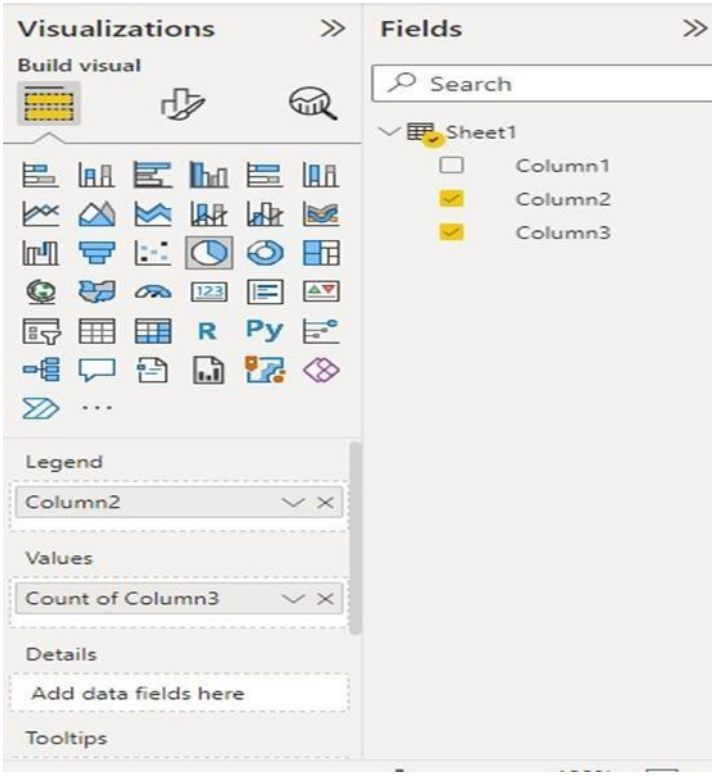
Column1	Column2	Column3
null	null	null
Median Income by Education Level	null	null
Education Level	Men	Women
No high school diploma	22138	13076
High school graduate	31683	20179
Some college or associate's degree	39601	25736
Bachelor's degree	53693	26250
Graduate or professional degree	71918	47319
null	null	null
Source: U.S. Census Bureau, 2005 American Community Survey	null	null
http://www.census.gov/prod/2006pubs/acs-02.pdf, page 13	null	null

Step 3:

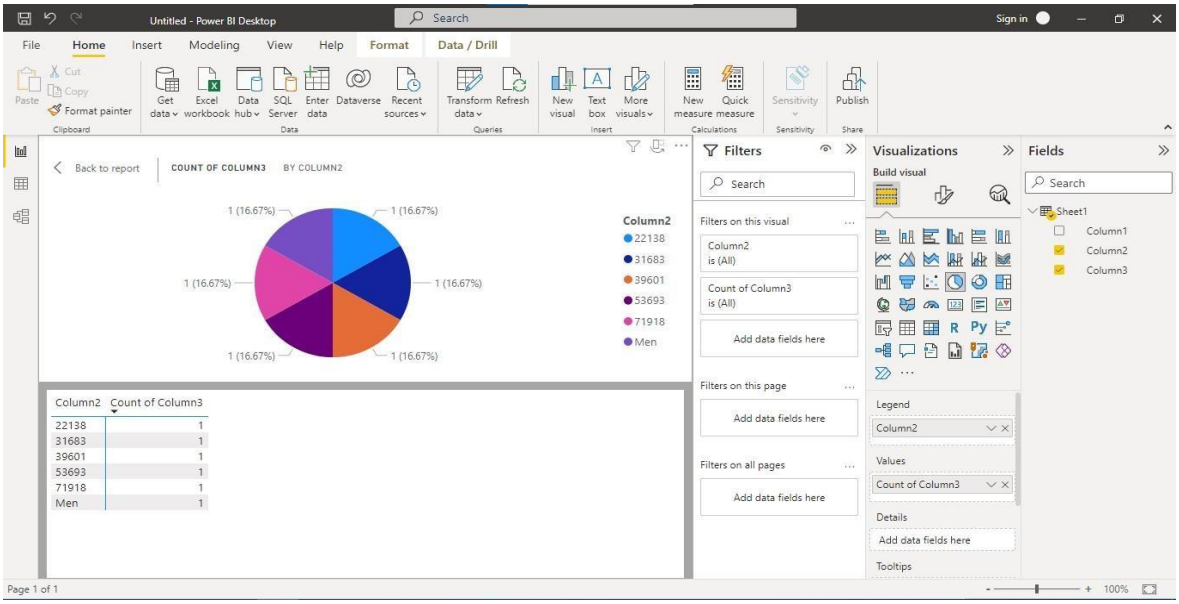
Now click on the columns that you want to see in the graphical manner. On left select the graph visualization



Step 4:
Now drag and drop the coloumns to **Legend** and **Values** option.

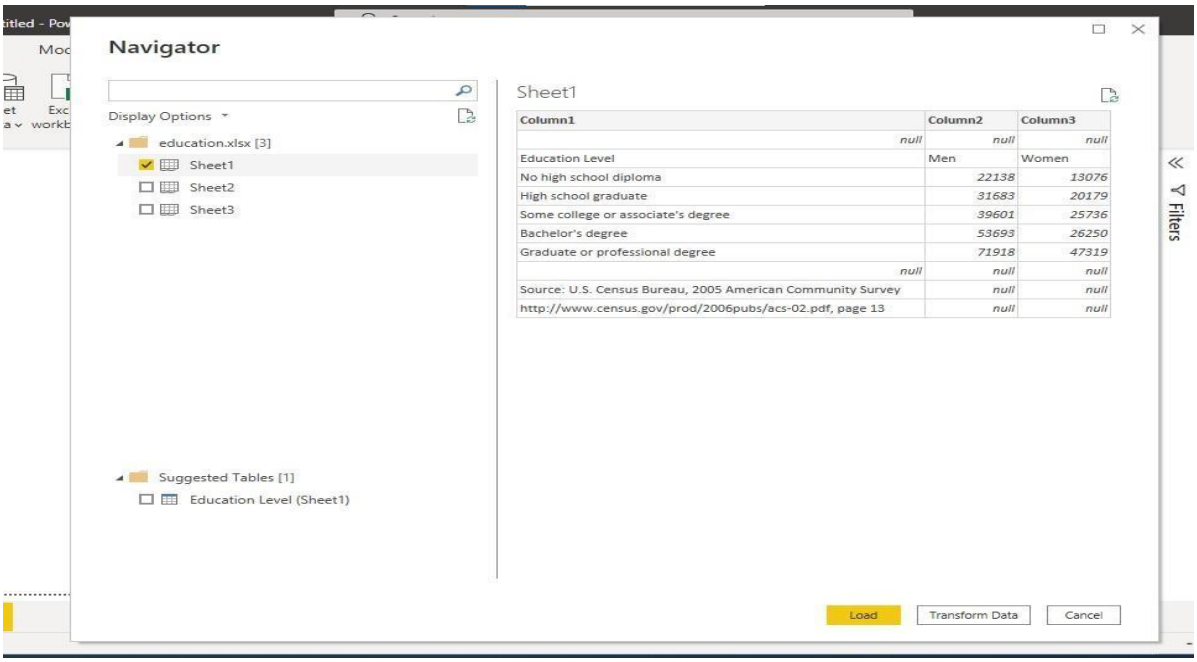


Step 5:
Here you will be able to see the graphical representation of **education.xls**

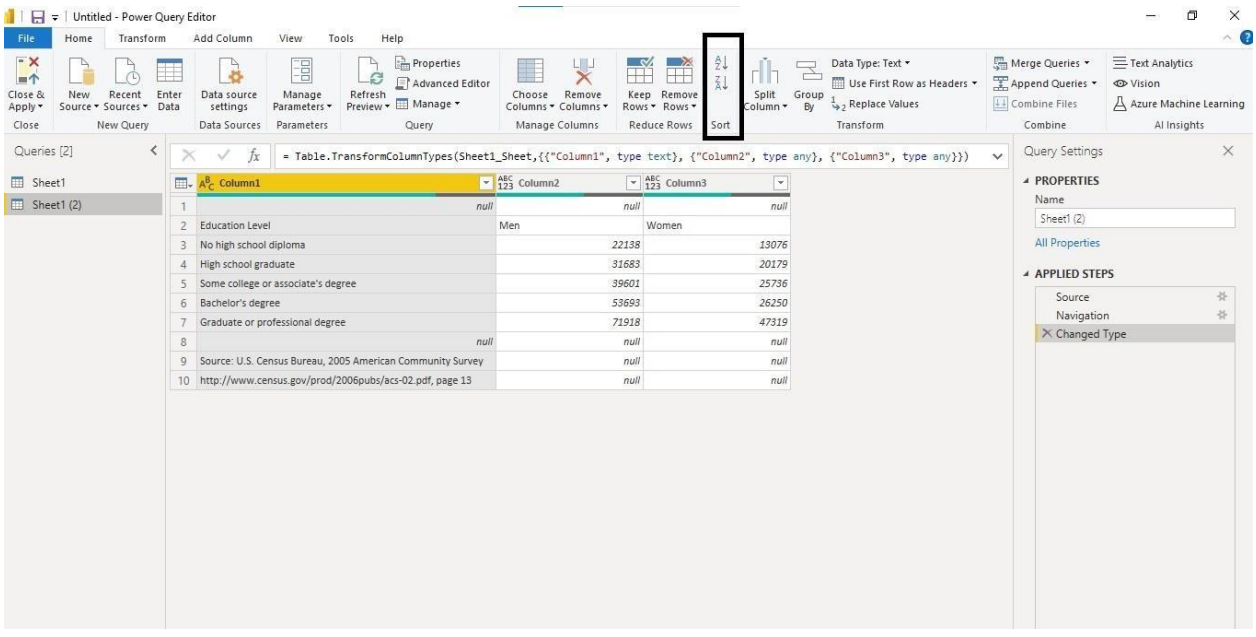


Now, Transfrom the data into the different formats

Step 1: Open the power B.I and import the data **education.xlsx.** Now click on sheet 1 and Transform the file option.

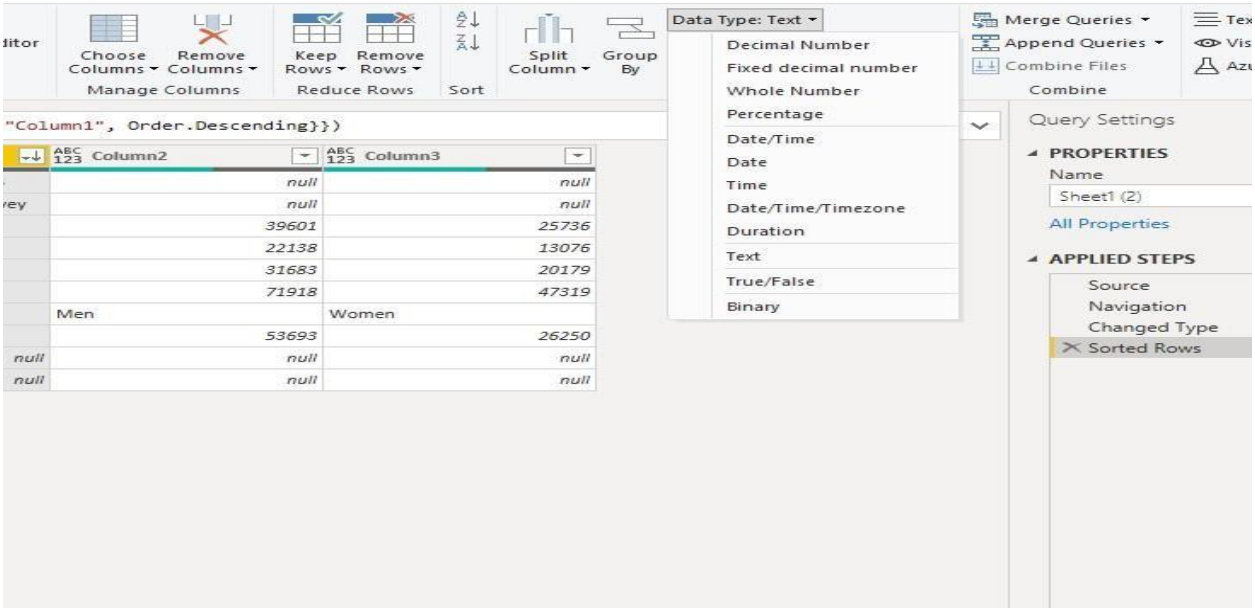


Step 2: Now Click on sort option for Asccending and Desending order



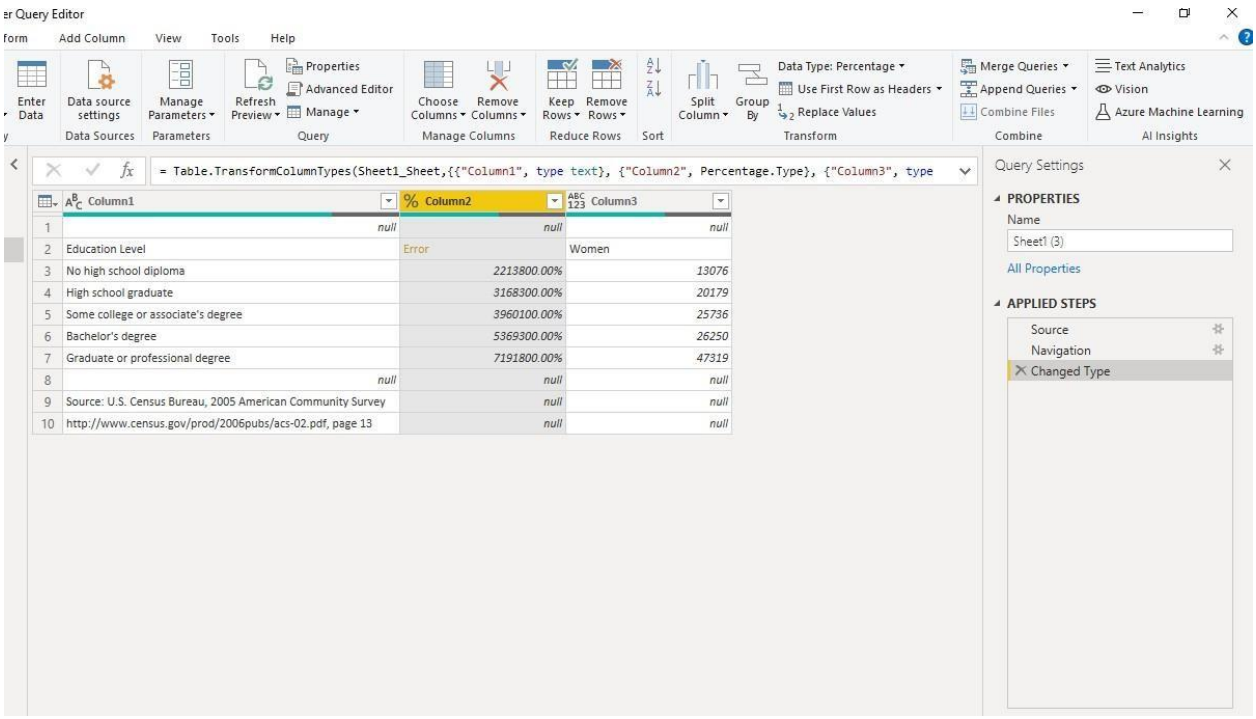
Step 3:

Now, Click on Data Type option. For different representation of data.



Step 4:

In Cloumn 2 you can see the percentage representation of data.



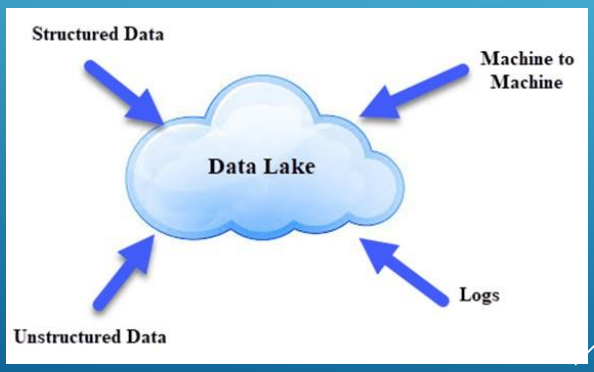
PRESENTATION

DATA LAKE

- Introduction of data lake: A Data Lake is storage repository of large amount of row data that means structure, semi-structure, unstructured data.
- This is the place where you can store three types of data structure, semi-structure, unstructured data with no fix amount of limit and storage to store the data.
- Data Lake follow to store less data into the structure database because it follows the schema on read process architecture to store the data.
- Data Lake allow us to transform the raw data that means structure, semi-structure, unstructured data into the structure data format so that SQL query could be performed for the analysis.

- Retrieval of data is so fast because there is no schema applied. Data must be access without any failure or any complex reason.
- Data Lake is similar to real time river or lake where the water comes from different-different places and at the last all the small-small river and lake are merged into the big river or lake where large amount of water are stored, whenever there is need of water then it can be used by anyone.
- It is low cost and effective way to store the large amount of data stored into centralized database for further organizational analysis and deployment.

DATA LAKE DIAGRAM



BUSINESS LAYER

Introduction of Business layer:

- The business layer is the transitional point between the nontechnical business requirements and desires and the practical data science, where, I suspect, most readers of this book will have a tendency to want to spend their careers, doing the perceived more interesting data science.
- The business layer is where we record the interactions with the business.
- This is where we convert business requirements into the data science requirements.

BUSINESS LAYER :

Contains the business requirements

1. Functional Requirements
2. Nonfunctional Requirements

The Functional Requirements

- Functional requirements record the detailed criteria that must be followed to realize the business's aspirations from its real-world environment when interacting with the data science ecosystem.
- These requirements are the business's view of the system.
- The MoSCoW method is a prioritization technique, to indicate how important each requirement is to the business.

Specific Functional Requirements

- The following requirements specific to data science environments will assist you in creating requirements that enable you to transform a business's aspirations into technical descriptive requirements.

Data Mapping Matrix

- The data mapping matrix is one of the core functional requirement recording techniques used in datascience.
- It tracks every data item that is available in the data sources.

Sun Models

- The sun models is a requirement mapping technique that assists you in recording requirements at a level that allows your nontechnical users to understand the intent of your analysis, while providing you with an easy transition to the detailed technical modeling of your data scientist and data engineer.
- Sun model supports three dimensions : person, location, and date.

Dimensions

- A dimension is a structure that categorizes facts and measures, to enable you to respond to businessquestions.
- A slowly changing dimension is a data structure that stores the complete history of the data loads in the dimension structure over the life cycle of the data lake.

Intra-Sun Model Consolidation Matrix

- The intra-sun model consolidation matrix is a tool that helps you to identify common dimensions between sun models

The Nonfunctional Requirements

Nonfunctional requirements record the precise criteria that must be used to appraise the operation of a data science ecosystem

Extensibility

- The ability to add extra features and carry forward customizations at next version upgrades within the data science ecosystem.
- The data science must always be capable of being extended to support new requirements.

Failure Management

- Failure management is the ability to identify the root cause of a failure and then successfully record all the relevant details for future analysis and reporting.

Fault Tolerance

- Fault tolerance is the ability of the data science ecosystem to handle faults in the system's processing.
- In simple terms, no single event must be able to stop the ecosystem from continuing the data science processing.

Latency

- Latency is the time it takes to get the data from one part of the system to another.

Interoperability

- Insist on a precise ability to share data between different computer systems under this section. Explain in detail what system must interact with what other systems

Maintainability

Insist on a precise period during which a specific component is kept in a specified state.