Practical Journal

DATA SCIENCE
SOFT COMPUTING TECHNIQUES

A Practical Report

Submitted in partial fulfillment of the

Requirements for the award of the Degree

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

Submitted by

Ms. Sokhi Rinki Jassi

Seat No: 1310236

DEPARTMENT OF INFORMATION TECHNOLOGY

VALIA C.L COLLEGE OF COMMERCE & VALIA L.C COLLEGE OF ARTS CES ROAD D.N NAGAR

(Affiliated to University of Mumbai)

MUMBAI, 400053

MAHARASHTRA

2023-2024

VALIA C.L COLLEGE OF COMMERCE & VALIA L.C COLLEGE OF ARTS CES ROAD D.N NAGAR

(Affiliated to University of Mumbai)

Mumbai-Maharashtra-400053

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the practical report “DATA SCIENCE” & “SOFT COMPUTING TECHNIQUES” is bonafide work of SOKHI RINKI JASSI bearing Seat No: 1310236 submitted in partial fulfilment of the requirements for the award of degree of MASTER OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide

Coordinator

External Examiner

Date:

College Seal

DATA SCIENCE

INDEX

PRAC NO.	PRACTICALS	DATE	SIGNATURE
1.	Creating Data Model using Cassandra.		
2.	Conversion from different formats to HORUS format. a. Text delimited CSV to HORUS format. b. XML to HORUS format. c. JSON to HORUS format. d. MySql Database to HORUS format. e. Picture(JPEG) to HORUS format. f. Video to HORUS format. g. Audio to HORUs format.		
3.	Utilities and Auditing. a. Fixers Utilities. b. Data Binning or Bucketing. c. Averaging of Data. d. Outlier Detection. e. Logging.		
4.	Retrieving Data.		
5.	Assessing Data.		
6.	Processing Data.		
7.	Transforming Data.		
8.	Organizing Data.		
9.	Generating Reports.		
10.	Data Visualization with Power BI.		

PRACTICAL 1

Aim :- Creating Data Model Using Cassandra

Steps:-
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)
```

PRACTICAL 2

Aim: Conversion from different formats to HORUS format.

A.Text delimited CSV to HORUS format

Code:-

```
# Utility Start CSV to HORUS
================================
# Standard Tools
#================================================

import pandas as pd

#Input   Agreement==================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'

InputData=pd.read_csv(sInputFileName,encoding="la n-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='C:/VKHCG/05-

DS/9999-Data/HORUS-CSV-Country.csv'

OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
        # Utility done ========================================
```

Output:-

```
Input Data Values ==================================
                       Country ISO-2-CODE ISO-3-Code ISO-M49  Unnamed: 4
0                     Malaysia         ML        MAL       4         NaN
1                        India         IN        IDN     248         NaN
2                  Los Angeles         LA        LAO       8         NaN
3                  Switzerland         SW        DZA      12         NaN
4                     New York         NY        ASM      16         NaN
..                         ...        ...        ...     ...         ...
242  Wallis and Futuna Islands         WF        WLF     876         NaN
243            Western Sahara         EH        ESH     732         NaN
244                     Yemen         YE        YEM     887         NaN
245                    Zambia         ZM        ZMB     894         NaN
246                  Zimbabwe         ZW        ZWE     716         NaN

[247 rows x 5 columns]
================================================
Process Data Values ================================
              CountryName  Unnamed: 4
CountryNumber
36                  tokyo         NaN
44                 israel         NaN
716              Zimbabwe         NaN
894                Zambia         NaN
887                 Yemen         NaN
...                   ...         ...
56                Belgium         NaN
112               Belarus         NaN
52               Barbados         NaN
50             Bangladesh         NaN
48                Bahrain         NaN

[247 rows x 2 columns]
================================================
CSV to HORUS - Done
Rinki_mscit
```

B. XML to HORUS Format

Code:-

```
# Utility Start XML to HORUS
==============================
# Standard Tools 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='C:/VKHCG/05-DS/9999-
Data/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='C:/VKHCG/05-
DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('================================================')
print('XML to HORUS - Done')
print('================================================')
# Utility done
================================================
```

Output:-

```
======================================================
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
Rinki_mscit
======================================================
```

C.JSON to HORUS Format

Code:-

```
# Utility Start JSON to HORUS ================================
```

```
# Standard Tools
#=================================================

import pandas as pd

# Input Agreement

=========================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.json'

InputData=pd.read_json(sInputFileName, orient='index', encoding="la n-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='c:/VKHCG/05-DS/9999-Data/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         WLE      716

[247 rows x 4 columns]
=====================================================
Process 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]
=====================================================
JSON to HORUS - Done
```

D.MySql Database to HORUS Format

Code:-

```
# Utility Start Database to HORUS
================================
# Standard Tools
#================================================

import pandas as pd import sqlite3 as sq

#Input  Agreement================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/ulity.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:/VKHCG/05-

DS/9999-Data/HORUS-CSV-Country.csv'

OutputData.to_csv(sOutputFileName, index = False)

print('Database to HORUS - Done')

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

Output:-

```
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
Rinki_mscit
```

E. Picture (JPEG) to HORUS Format

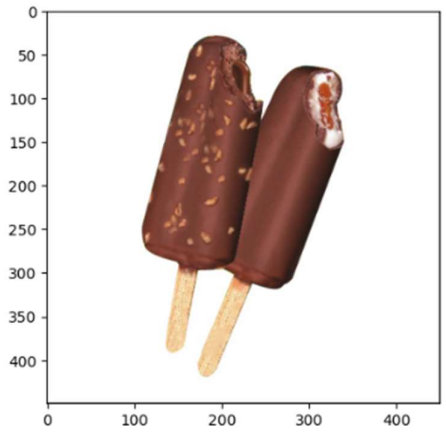
Code:-

```
from skimage import io import

pandas as pd import

matplotlib.pyplot as plt import

numpy as np


# Input Agreement sInputFileName='C:/VKHCG/05-DS/9999-

Data/ice-cream.jpg'

InputData = io.imread(sInputFileName, pilmode='RGBA')

plt.imshow(InputData)


InputData. shape
```

```
print('Input Data Values')
 print('X: ',InputData.shape[0])
 print('Y: ', InputData. shape[1])
print('RGBA: ',
InputData.shape[2])

ProcessRawData=InputData.flatten(
)     y=InputData.shape[2]     +     2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
ProcessRawData

ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
sColumns= [ 'XAxis', 'YAxis', 'Red', 'Green', 'Blue','Aplha']
ProcessData.columns=sColumns
ProcessData
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])

OutputData = ProcessData
OutputData.to_csv('Image to HORUS.csv', index = False)
```

Output:

```
Input Data Values
X:  450
Y:  450
RGBA:  4
Rows:  135000
Columns : 6
Rinki_mscit
```



F. Video to HORUS Format

Code:-

```
import os import shutil
import cv2
sInputFileName="C:/VKHCG/05-DS/9999Data/dog.mp4 "
sDataBaseDir="C:/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:
        break
count += 1
print('================================================')
print('Generated : ', count, ' Frames')
```

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

Output:-

G. Audio to HORUS Format

Code:-

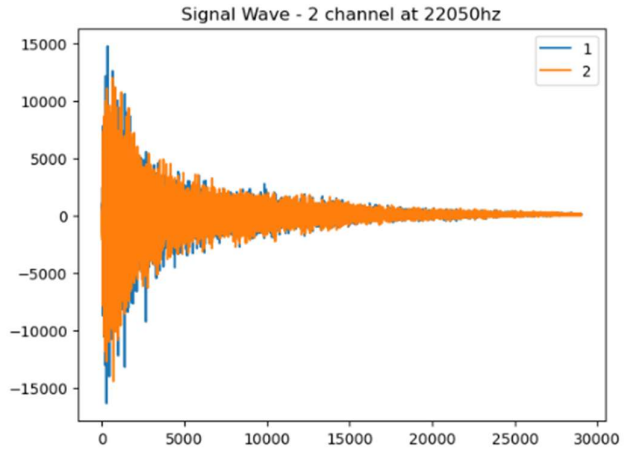
```
# Utility Start Audio to HORUS ================================
# Standard Tools
#=============================================================
from scipy.io import wavfile

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)
```

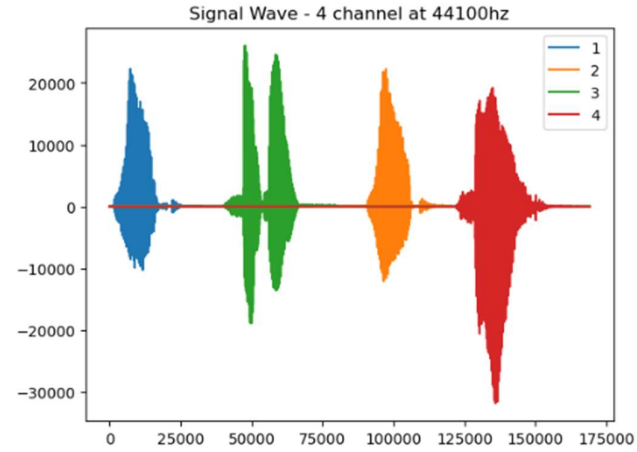
```
#=========================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('=========================================================')
print('Processing : ', sInputFileName)
print('=========================================================')
InputRate, 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/9999Data/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

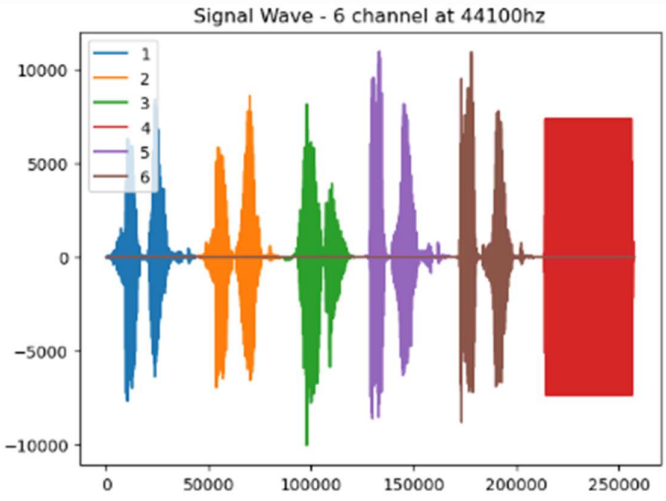
```
======================================================
Processing :  C:/VKHCG/05-DS/9999-Data/2ch-sound.wav
======================================================
----------------
Audio: 2 channel
----------------
Rate: 22050
----------------
shape: (29016, 2)
dtype: int16
min, max: -16384 14767
----------------
```



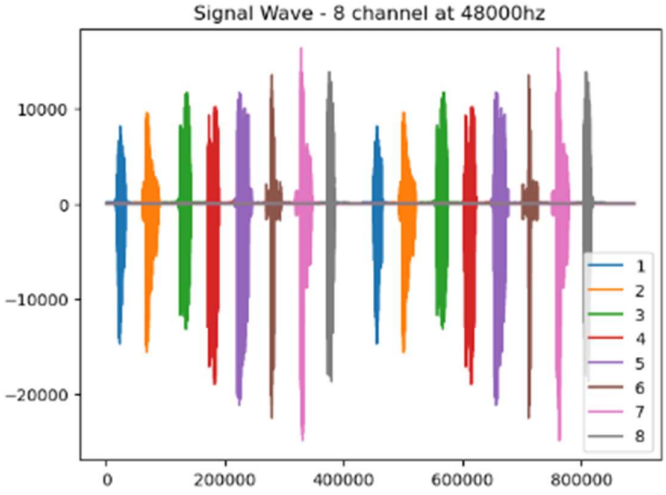

```
======================================================
Processing :  C:/VKHCG/05-DS/9999-Data/4ch-sound.wav
======================================================
----------------
Audio: 4 channel
----------------
Rate: 44100
----------------
shape: (169031, 4)
dtype: int16
min, max: -31783 26018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




```
======================================================
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
---------------
```




```
==================================================
Audio to HORUS - Done
Rink_mscit
==================================================
```

PRACTICAL 3

Aim: Utilities and Auditing

A.Fixers Utilities:

 # 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,'<')
```

Output:-

 # 2 Removing nonprintable characters from a data entry

```
import string
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

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

Output:-

```
print('Rinki sokhi practical 3')
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,'<')
import string
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)
import datetime
print('# 3 Reformatting data entry to match specific formatting criteria.')
baddate = datetime.date(2019, 10, 31)
baddata=format(baddate,'%Y-%m-%d')
gooddate = datetime.datetime.strptime(baddata,'%Y-%m-%d')
gooddata=format(gooddate,'%d %B %Y')
print('Bad Data : ',baddata)
print('Good Data : ',gooddata)
```

```
Rinki sokhi practical 3
#1 Removing leading or lagging spaces from a data entry
>  Data Science with too many spaces is bad!!!  <
> Data Science with too many spaces is bad!!! <
#2 Removing nonprintable characters from a data entry
Bad Data :  DataScience with funny characters is bad!!!
Clean Data :  DataScience with funny characters is bad!!!
# 3 Reformatting data entry to match specific formatting criteria.
Bad Data :  2019-10-31
Good Data :  31 October 2019
```

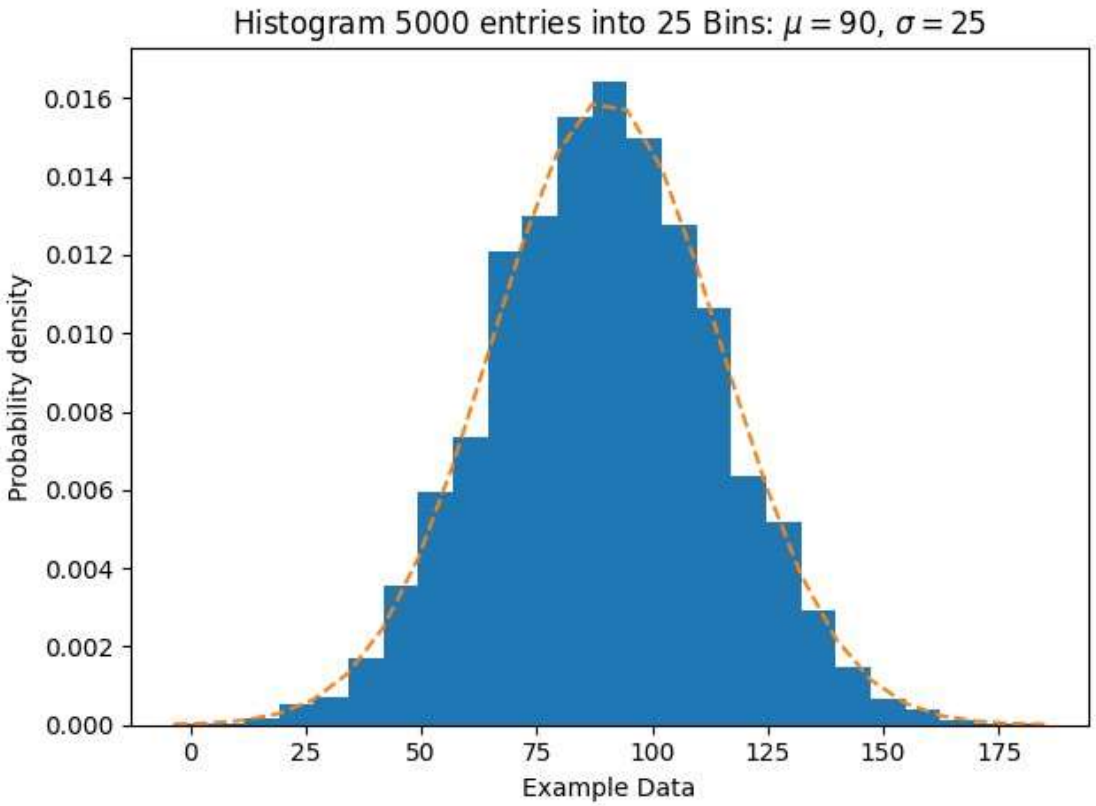
B. Data Binning or Bucketing

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: $\mu=' + str(mu)
+ '$,
$\sigma=' +str(sigma) + '$'
ax.set_title(sTitle)
fig.tight_layout()
sPathFig='C:/VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png'
```

Output:-



C. Averaging of Data

Code:

```
import pandas as pd
#################################################################
InputFileName='IP_DATA_CORE.csv'
```

```
OutputFileName='Retrieve_Router_Loca on.csv'

Base='C:/VKHCG'

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

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

sFileName=Base + '/01-Vermeulen/00-RawData/' +

InputFileName print('Loading :',sFileName)

IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,usecols=['Country','
Place Name','La tude','Longitude'], encoding="la n-1")
IP_DATA_ALL.rename(columns={'Place      Name':      'Place_Name'},

inplace=True)  AllData=IP_DATA_ALL[['Country',  'Place_Name','La

tude']] print(AllData)

MeanData=AllData.groupby(['Country', 'Place_Name'])['La tude'].mean()

print(MeanData)

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

Output:-

```
Working Base : Rinki_mscit/VKHCG
##############################
Loading : Rinki_mscit/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
     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

Code:-

```
import pandas as pd
import sys
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv' if sys.platform ==
'linux':
   Base=os.path.expanduser('~') + '/ASUS/DESKTOP' else:
   Base="C:\VKHCG" print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sFileName=Base + '/Asus/Desktop/' + InputFileName print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,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)
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:-

```
PS C:\Users\asus\Desktop\Rinki_Sokhi> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.e
xe c:/Users/asus/Desktop/Rinki Sokhi/demo.py
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
```

E. Logging

Code:-

```
import sys
import os
import logging
 import uuid
import shutil
import time
 if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG' else:
    Base='C:/VKHCG'
sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04Clark']
sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05Organise','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()
for hdlr in log.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.DEbG,
    format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
                        datefmt='%m-%d %H:%M',
                        filename=sLogFile,
                        filemode='w')
       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\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop/Rinki_mscit/VKHCG/77-Yoke/Yok
e_Logging.py
Set up: C:/VKHCG/01-Vermeulen/01-Retrieve/Logging/Logging_bd924eac-0e8a-45b8-9dff-544455141089.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_10cbfd2d-d9a2-45e3-a951-981852593fd4.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_1075cf6e-1568-4797-9f2c-ae58d540278d.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_4dd41797-e153-42f0-8539-f1a2a7c74fab.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_7993a390-eaac-4c69-838a-5751003ee222.log
root        : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-05-Organise-info: INFO     Practical Data Science logged information message.
```

PRACTICAL 4

Retrieve Superstep

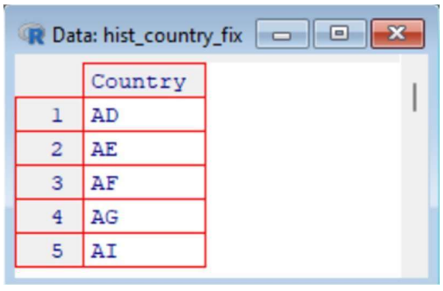
A. Program to retrieve and perform data processing using R

```
[Previously saved workspace restored]

> library(readr)
> IP_DATA_ALL <- read_csv("C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
New names:
• `` -> `...1`
Rows: 1048575 Columns: 9
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (3): Country, Place.Name, Post.Code
dbl (6): ...1, ID, Latitude, Longitude, First.IP.Number, Last.IP.Number

ⓘ Use `spec()` to retrieve the full column specification for this data.
ⓘ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> cols(
+ ID = col_double(), Country = col_character(),
+ `Place Name` = col_character(),
+ `Post Code` = col_double(), Latitude = col_double(), Longitude = col_double(),
+ `First IP Number` = col_double(),
+ `Last IP Number` = col_double()
+ )
cols(
  ID = col_double(),
  Country = col_character(),
  `Place Name` = col_character(),
  `Post Code` = col_double(),
  Latitude = col_double(),
  Longitude = col_double(),
  `First IP Number` = col_double(),
  `Last IP Number` = col_double()
)
> View(IP_DATA_ALL)
```

Data: IP_DATA_ALL

	...1	ID	Country
1	1	1	BW
2	2	2	BW
3	3	3	BW
4	4	4	BW
5	5	5	BW
6	6	6	BW
7	7	7	BW
8	8	8	BW

```
> spec(IP_DATA_ALL)
cols(
  ...1 = col_double(),
  ID = col_double(),
  Country = col_character(),
  Place.Name = col_character(),
  Post.Code = col_character(),
  Latitude = col_double(),
  Longitude = col_double(),
  First.IP.Number = col_double(),
  Last.IP.Number = col_double()
)
> library(tibble)
> set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)
# A tibble: 1,048,575 × 9
    ...1     ID Country Place.Name Post.Code Latitude Longitude First.IP.Number
   <dbl> <dbl> <chr>   <chr>      <chr>        <dbl>     <dbl>           <dbl>
 1     1     1 BW      Gaborone   <NA>         -24.6      25.9       692781056
 2     2     2 BW      Gaborone   <NA>         -24.6      25.9       692781824
 3     3     3 BW      Gaborone   <NA>         -24.6      25.9       692909056
 4     4     4 BW      Gaborone   <NA>         -24.6      25.9       692909568
 5     5     5 BW      Gaborone   <NA>         -24.6      25.9       693051392
 6     6     6 BW      Gaborone   <NA>         -24.6      25.9       693059584
 7     7     7 BW      Gaborone   <NA>         -24.6      25.9       693059840
 8     8     8 BW      Gaborone   <NA>         -24.6      25.9       693060096
 9     9     9 BW      Gaborone   <NA>         -24.6      25.9       693060352
10    10    10 BW      Gaborone   <NA>         -24.6      25.9       693060608
# ℹ 1,048,565 more rows
# ℹ 1 more variable: Last.IP.Number <dbl>
# ℹ Use `print(n = ...)` to see more rows
> IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL, syntacti
> sapply(IP_DATA_ALL_FIX, typeof)
       ...1          ID     Country  Place.Name   Post.Code    Latitude
   "double"    "double"  "character"  "character"  "character"  "double"
  Longitude First.IP.Number  Last.IP.Number
   "double"      "double"       "double"
> library(data.table)
data.table 1.14.8 using 2 threads (see ?getDTthreads).  Latest news: r-datatable.com
> hist_country=data.table(Country=unique(IP_DATA_ALL_FIX[is.na(IP_DATA_ALL_FIX ['Country']) == 0, ]$Country
+ ))
> setorder(hist_country,'Country')
> hist_country_with_id=rowid_to_column(hist_country, var = "RowIDCountry")
> hist_country_fix=set_tidy_names(hist_country, syntactic = TRUE, quiet = TRUE)
> View(hist_country_fix)
```



	Country
1	AD
2	AE
3	AF
4	AG
5	AI

```
> IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_FIX, table(Country)))
> View(IP_DATA_COUNTRY_FREQ)
```

Data: IP_DATA_COUNTRY_FREQ

	Country	N
1	AD	46
2	AE	1735
3	AF	15
4	AG	15
5	AI	9
6	AL	88
7	AM	85
8	AO	98

```
>  hist_latitude = data.table(Latitude=unique(IP_DATA_ALL_FIX [is.na(IP_DATA_ALL_FIX['Latitude']) == 0,]$Latitude))
> setkeyv(hist_latitude, 'Latitude')
> setorder(hist_latitude)
> hist_latitude_with_id=rowid_to_column(hist_latitude, var = "RowID")
> View(hist_latitude_with_id)
```

Data: hist_latitude_with_id

	RowID	Latitude
1	1	-54.1561
2	2	-51.7000
3	3	-49.3500
4	4	-46.6333
5	5	-46.4382
6	6	-46.4308
7	7	-46.4000

```
> IP_DATA_Latitude_FREQ=data.table(with(IP_DATA_ALL_FIX,table(Latitude)))
> View(IP_DATA_Latitude_FREQ)
```

Data: IP_DATA_Latitude_FREQ

	Latitude	N
1	-54.1561	1
2	-51.7	2
3	-49.35	2
4	-46.6333	1
5	-46.4382	8
6	-46.4308	2
7	-46.4	3

```
> sapply(IP_DATA_ALL_FIX[,'Latitude'], min, na.rm=TRUE)
Latitude
-54.1561
> sapply(IP_DATA_ALL_FIX[,'Country'], min, na.rm=TRUE)
Country
   "AD"
> sapply(IP_DATA_ALL_FIX[,'Latitude'], max, na.rm=TRUE)
Latitude
 78.2167
> sapply(IP_DATA_ALL_FIX[,'Country'], max, na.rm=TRUE)
Country
   "ZW"
> sapply(IP_DATA_ALL_FIX [,'Latitude'], mean, na.rm=TRUE)
Latitude
39.28514
> sapply(IP_DATA_ALL_FIX [,'Latitude'], median, na.rm=TRUE)
Latitude
 41.4406
> sapply(IP_DATA_ALL_FIX [,'Latitude'], range, na.rm=TRUE)
     Latitude
[1,] -54.1561
[2,]  78.2167
> sapply(IP_DATA_ALL_FIX [,'Latitude'], quantile, na
      Latitude
0%    -54.1561
25%    35.8204
50%    41.4406
75%    48.5613
100%   78.2167
> sapply(IP_DATA_ALL_FIX [,'Latitude'], sd, na.rm=TR
Latitude
15.88847
```

```
> sapply(IP_DATA_ALL_FIX [,'Longitude'], sd, na.rm=TRUE)
Longitude
  68.0561
>
```

B.Program to retrieve different attributes of data.

Code:-

```
##############Retrieve-IP_DATA_ALL.py#######################
# -*- coding: utf-8 -*-
#####################################################################
import sys import os import pandas as pd
#####################################################################
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="latin1")
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

Output:-

```
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 1048575
Columns: 9
### Raw Data Set ####################################
Unnamed: 0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
### Fixed Data Set ##################################
Unnamed:.0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
### Done!! ##########################################
```

C.Program to create data patterns

Creating a Data pattern by taking a string input and creating a pattern of alternative uppercase and lowercase

Code:-

```
import pandas as pd


 test_str = "I like Butter Chicken"

# printing original string

print("The original string is : " + str(test_str))


# Using upper() + lower() + loop

# Alternate cases in String

res = ""
```

```
for idx in range(len(test_str)):
    if not idx % 2:
        res = res + test_str[idx].upper()
    else:
        res = res + test_str[idx].lower()

# printing result
print("The alternate case string is : " + str(res))
```

Output:-

```
import pandas as pd

test_str = "I like Butter Chicken"
# printing original string
print("The original string is : " + str(test_str))

# Using upper() + lower() + loop
# Alternate cases in String
res = ""
for idx in range(len(test_str)):
    if not idx % 2:
        res = res + test_str[idx].upper()
    else:
        res = res + test_str[idx].lower()

# printing result
print("The alternate case string is : " + str(res))
```

```
The original string is : I like Butter Chicken
The alternate case string is : I LiKe bUtTeR ChIcKeN
```

D. Loading IP_DATA_ALL.

Code:

```
import sys import os

import pandas as pd

######################################################################
Base='C:/VKHCG'
######################################################################

sFileName=Base + '/01-Vermeulen/00-RawData/Created_data.csv'

print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,

encoding="la                                                                    n-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="la n-1")

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

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

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

Output:-

```
PS C:\Users\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop/
Rinki_mscit/VKHCG/01-Vermeulen/01-Retrieve/Retrieve-IP_DATA_ALL.py
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 3562
Columns: 8
### Raw Data Set #################################
ID <class 'str'>
Country <class 'str'>
Place Name <class 'str'>
Post Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First IP Number <class 'str'>
Last IP Number <class 'str'>
### Fixed Data Set ###############################
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
### Done!! ########################################
```

PRACTICAL 5

Assessing Data.

A.Drop the Columns Where All Elements Are Missing Values.

Code:

```
############################################################
########
# -*- coding: utf-8 -*-
############################################################
######## import sys
import os import panda
as pd
############################################################
########
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG' else:
   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\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop/Rinki_mscit/VKHCG/01-Vermeule
n/02-Assess/Assess-Good-Bad-01.py
##############################
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
```

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

Code:

```
#############################################################
########
# -*- coding: utf-8 -*- ###############################################################
######## 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'
#############################################################
########
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG' else:
   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('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\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop/Rinki_mscit/VKHCG/01-Vermeule
n/02-Assess/Assess-Good-Bad-02.py
################################
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
```

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

Code:

```
################################################################
########
# -*- coding: utf-8 -*-
################################################################
######## import sys
import os
import pandas as pd
################################################################
########
Base='C:/VKHCG' sInputFileName='Good-or-Bad.csv' sOutputFileName='Good-or-Bad-
03.csv'
Company='01-Vermeulen'
################################################################
########
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG' else:
    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('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:-

```
>
======RESTART:C:/VKHCG/01 -Vermulen /02 – Assess/ Assess-Good-Bad-02.y=======
################################
Working Base : C:/VKHCG using win32
########################## #####
Loading:C :/ VKHCG/01-Vermulen 00/ -RawData/Good-or-Bad csv
################################
##############################
## Raw Data Values
##############################
         version https://git-lfs.github.com/spec/v1
0  oid sha256:40efd836a4bff3496f163848ea56eff65ab...
1                                          size 576
##############################
## Data Profile
##############################
Rows : 2
Columns : 1
##############################
##############################
## Test Data Values
##############################
         version https://git-lfs.github.com/spec/v1
0  oid sha256:40efd836a4bff3496f163848ea56eff65ab...
1                                          size 576
##############################
## Data Profile
##############################
Rows : 2
Columns : 1
##############################
##############################
### Done!! ####################
##############################
```

PRACTICAL 6

Aim: Processing Data.

A. Build the time hub, links, and satellites.

Code:

```
############################################################
########
# -*- coding: utf-8 -*-
############################################################
######## import sys
import os from datetime
import datetime from datetime
import timedelta from pytz
import timezone, all_timezones
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='01-Vermeulen'
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)
############################################################
########
base = datetime(2018,1,1,0,0,0)
numUnits=10*365*24
############################################################
########
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)] t=0 for i in date_list:
   now_utc=i.replace(tzinfo=timezone('UTC'))      sDateTime=now_utc.strftime("%Y-%m-
%d %H:%M:%S")       print(sDateTime)
   sDateTimeKey=sDateTime.replace(' ','-').replace(':','-')     t+=1
   IDNumber=str(uuid.uuid4())
   TimeLine=[('ZoneBaseKey', ['UTC']),
          ('IDNumber', [IDNumber]),
          ('nDateTimeValue', [now_utc]),
          ('DateTimeValue', [sDateTime]),
('DateTimeKey', [sDateTimeKey])]      if t==1:
      TimeFrame = pd.DataFrame(columns = ['ZoneBaseKey' ,
'IDNumber' , 'nDateTimeValue' , 'DateTimeValue' , 'DateTimeKey'])      else:
       TimeRow = pd.DataFrame(columns = ['ZoneBaseKey' ,
'IDNumber' , 'nDateTimeValue' , 'DateTimeValue' , 'DateTimeKey'])
       TimeFrame = pd.concat([TimeFrame,TimeRow])
############################################################ ########
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','D ateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
############################################################
########
TimeFrame.set_index(['IDNumber'],inplace=True)
############################################################
######## sTable = 'Process-
Time' print('Storing
:',sDatabaseName,'
Table:',sTable)
TimeHubIndex.to_sql(sTable,
conn1, if_exists="replace")
############################################################
######## sTable = 'Hub-
Time'
print('Storing :',sDatabaseName,' Table:',sTable) TimeHubIndex.to_sql(sTable, conn2,
if_exists="replace")
############################################################
########
```

```
active_timezones=all_timezones z=0 for
zone in active_timezones:
   t=0     for j in range(TimeFrame.shape[0]):
now_date=TimeFrame['nDateTimeValue'][j]
DateTimeKey=TimeFrame['DateTimeKey'][j]
now_utc=now_date.replace(tzinfo=timezone('UTC'))
sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S")          now_zone =
now_utc.astimezone(timezone(zone))
sZoneDateTime=now_zone.strftime("%Y-%m-%d
%H:%M:%S")
      print(sZoneDateTime)        t+=1
      z+=1
      IDZoneNumber=str(uuid.uuid4())
      TimeZoneLine=[('ZoneBaseKey', ['UTC']),
                ('IDZoneNumber', [IDZoneNumber]),
                ('DateTimeKey', [DateTimeKey]),
                ('UTCDateTimeValue', [sDateTime]),
                ('Zone', [zone]),
                ('DateTimeValue', [sZoneDateTime])]         if t==1:
        TimeZoneFrame = pd.DataFrame(columns = ['ZoneBaseKey',
'IDZoneNumber' , 'DateTimeKey' , 'UTCDateTimeValue' , 'Zone' , 'DateTimeValue'])
      else:
         TimeZoneRow = pd.DataFrame(columns = ['ZoneBaseKey',
'IDZoneNumber' , 'DateTimeKey' , 'UTCDateTimeValue' , 'Zone' ,
'DateTimeValue'])
         TimeZoneFrame =
pd.concat([TimeZoneFrame,TimeZoneRow])


TimeZoneFrameIndex=TimeZoneFrame.set_index(['IDZoneNumber']
,inplace=False)
   sZone=zone.replace('/','-').replace(' ','')

############################################################
#####
   sTable = 'Process-Time-'+sZone
   print('Storing :',sDatabaseName,' Table:',sTable)
   TimeZoneFrameIndex.to_sql(sTable, conn1, if_exists="replace")
############################################################
#########

############################################################
#####
   sTable = 'Satellite-Time-'+sZone
   print('Storing :',sDatabaseName,' Table:',sTable)
```

```
    TimeZoneFrameIndex.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:-

```
2008-01-05 03:00:00
2008-01-05 02:00:00
2008-01-05 01:00:00
2008-01-05 00:00:00
2008-01-04 23:00:00
2008-01-04 22:00:00
2008-01-04 21:00:00
2008-01-04 20:00:00
2008-01-04 19:00:00
2008-01-04 18:00:00
2008-01-04 17:00:00
2008-01-04 16:00:00
2008-01-04 15:00:00
2008-01-04 14:00:00
2008-01-04 13:00:00
2008-01-04 12:00:00
2008-01-04 11:00:00
2008-01-04 10:00:00
2008-01-04 09:00:00
2008-01-04 08:00:00
2008-01-04 07:00:00
2008-01-04 06:00:00
2008-01-04 05:00:00
2008-01-04 04:00:00
2008-01-04 03:00:00
2008-01-04 02:00:00
2008-01-04 01:00:00
Traceback (most recent call last):
  File "c:\Users\asus\Desktop\Rinki_mscit\VKHCG\01-Vermeulen\03-Process\Process_Time.py",
```

B. Golden Nominal.

Code:

```
############################################################
######## 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','LastNa me','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, conn2,
if_exists="replace")
############################################################ ########
PersonSatelliteGenderRaw=Data[['PersonID','FirstName','SecondNam e','LastName'\
                    ,'BirthDateKey','Gender']]
PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lamb da row:
            str(uuid.uuid4())
            ,axis=1)
PersonSatelliteGender=PersonSatelliteGenderRaw.drop_duplicates(su bset=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, conn2,
if_exists="replace")
############################################################ ########
PersonSatelliteBirthdayRaw=Data[['PersonID','FirstName','SecondNa me','LastName',\
                     'BirthDateKey','TimeZone','BirthDate']]
PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lam bda 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, conn2,
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) sql.execute(sSQL,conn2)
print('#################')
#############################################################
###
######### print('###
Done!!
################################################')
#############################################################
#########
```

Output:-

```
PS C:\Users\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop/Rinki_mscit/VKHCG/04-Clark/03
-Process/Process-People.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. Vehicles.

Code:

```
#############################################################
########
# -*- 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[['Make','Model']].copy()
VehicleKey=VehicleRawKey.drop_duplicates()
```

```
############################################################
########
VehicleKey['ObjectKey']=VehicleKey.apply(lambda row:
   str('('+ str(row['Make']).strip().replace(' ', '-').replace('/', '-').lower() +
    ')-(' + (str(row['Model']).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
############################################################
########
#
VehicleHub=VehicleKey[['ObjectType','ObjectKey','ObjectUUID']].c opy()
VehicleHub.index.name='ObjectHubID' sTable = 'Hub-Object-Vehicle'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleHub.to_sql(sTable, conn2, if_exists="replace")
############################################################
########
### Vehicle Satellite
############################################################
########
#
VehicleSatellite=VehicleKey[['ObjectType','ObjectKey','ObjectUUID'
,'Make','Model']].copy()
VehicleSatellite.index.name='ObjectSatelliteID' sTable = 'Satellite-
Object-Make-Model' print('Storing :',sDatabaseName,' Table:',sTable)
VehicleSatellite.to_sql(sTable, conn2, 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,conn2) 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,
conn2)

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) sql.execute(sSQL,conn2)
print('################')
############################################################
######### conn1.close()
conn2.close()
############################################################ #########
#print('### Done!!
################################################')
############################################################ #########
```

Output:-

```
PS C:\Users\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop/
Rinki_mscit/VKHCG/03-Hillman/03-Process/Process-Vehicle-Logistics.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
       Cylinders  EngineSize                DriveWheels  FuelCostperYear  \
0            4.0         2.0           Rear-Wheel Drive             1900
1           12.0         4.9           Rear-Wheel Drive             3600
2            4.0         2.2          Front-Wheel Drive             1450
3            8.0         5.2           Rear-Wheel Drive             3600
4            4.0         2.2  4-Wheel or All-Wheel Drive             2500
...          ...         ...                        ...              ...
39096        4.0         2.2          Front-Wheel Drive             1800
39097        4.0         2.2          Front-Wheel Drive             1700
39098        4.0         2.2  4-Wheel or All-Wheel Drive             1900
39099        4.0         2.2  4-Wheel or All-Wheel Drive             1900
39100        4.0         2.2  4-Wheel or All-Wheel Drive             2600

              FuelType        Make                Model     Transmission  \
0      Regular Gasoline  Alfa Romeo   Spider Veloce 2000     Manual 5-spd
1      Regular Gasoline     Ferrari            Testarossa     Manual 5-spd
2      Regular Gasoline       Dodge               Charger     Manual 5-spd
3      Regular Gasoline       Dodge  B150/B250 Wagon 2WD  Automatic 3-spd
4      Premium Gasoline      Subaru     Legacy AWD Turbo     Manual 5-spd
...                 ...         ...                  ...              ...
39096  Regular Gasoline      Subaru              Legacy  Automatic 4-spd
39097  Regular Gasoline      Subaru              Legacy     Manual 5-spd
39098  Regular Gasoline      Subaru          Legacy AWD  Automatic 4-spd
39099  Regular Gasoline      Subaru          Legacy AWD     Manual 5-spd
39100  Premium Gasoline      Subaru     Legacy AWD Turbo  Automatic 4-spd
```

D. Human-Environment Interaction.

Code:

```
################################################################
########
# -*- 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','Lon gitude','Latitude'])
else:
         LocationRow = pd.DataFrame(columns =
['ObjectBaseKey','IDNumber','LocationNumber','LocationName','Lon gitude','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:-

```
PS C:\Users\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop/
Rinki_mscit/VKHCG/01-Vermeulen/03-Process/Process_Location.py
################################
Working Base : C:/VKHCG  using  win32
################################
Create: 1  of  64800 : L-180000--090000
Create: 2  of  64800 : L-180000--080000
Create: 3  of  64800 : L-180000--070000
Create: 4  of  64800 : L-180000--060000
Create: 5  of  64800 : L-180000--050000
Create: 6  of  64800 : L-180000--040000
Create: 7  of  64800 : L-180000--030000
Create: 8  of  64800 : L-180000--020000
Create: 9  of  64800 : L-180000--010000
Create: 10  of  64800 : L-180000-+000000
Create: 11  of  64800 : L-180000-+010000
Create: 12  of  64800 : L-180000-+020000
Create: 13  of  64800 : L-180000-+030000
Create: 14  of  64800 : L-180000-+040000
Create: 15  of  64800 : L-180000-+050000
Create: 16  of  64800 : L-180000-+060000
Create: 17  of  64800 : L-180000-+070000
Create: 18  of  64800 : L-180000-+080000
Create: 19  of  64800 : L-170000--090000
Create: 20  of  64800 : L-170000--080000
```

E. Forecasting.

Code:

```
############################################################
######## 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\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop
Rinki_mscit/VKHCG/01-Vermeulen/03-Process/Process_Location.py
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/04-Clark/00-RawData/VKHCG_Shares.csv
################################
Rows   : 2
Columns: 1
################
################################
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
################################
################################
```

PRACTICAL 7

Aim: Transform Data.

A.Transform Superstep.

Code:-

```
############################################################
########
# -*- 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','Zo ne','DateTimeValue'])
############################################################ ########
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','D ateTimeValue']]
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','DateTi meValue']]
TimeSatelliteIndex=TimeSatellite.set_index(['IDNumber'],inplace=Fa lse)
############################################################
########
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:-

```
PS C:\Users\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/asus/Desktop/
Rinki_mscit/VKHCG/01-Vermeulen/04-Transform/Transform-Gunnarsson_is_Born.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)
################################
```

PRACTICAL 8

Aim: Organizing Data.

A.Horizontal Style.

Code:-

```
    ############################################################
########
# -*- 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, conn1)
#############################################################
########
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, conn1)
#############################################################
########
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\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/User
s/asus/Desktop/Rinki_mscit/VKHCG/01-Vermeulen/05-Organise/Organize-Horizontal.py
##############################
Working Base : C:/VKHCG  using  win32
##############################
Row: 1 of 1080.0
Traceback (most recent call last):
  File "c:\Users\asus\Desktop\Rinki_mscit\VKHCG\01-Vermeulen\05-Organise\Organize-Horizontal
.py", line 66, in <module>
    PersonFrame = pd.DataFrame.from_items(PersonLine)
                  ^^^^^^^^^^^^^^^^^^^^^^
AttributeError: type object 'DataFrame' has no attribute 'from_items'
PS C:\Users\asus>
```

B. Vertical Style.

Code:

```
import sys
import os
import pandas
as pd
import sqlite3 as sq
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
Company='01-Vermeulen' sDataWarehouseDir=Base +
'/99-DW' 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, conn1)
#############################################################
########
print('##################################')
sTable = 'Dim-BMI' print('Loading
:',sDatabaseName,' Table:',sTable)
print('##################################')
sSQL="SELECT\
   Height,\
   Weight,\
   Indicator\
   FROM[Dim-BMI];"
PersonFrame1=pd.read_sql_query(sSQL,conn1)
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
sTable='Dim-BMI-Vertical' print('\n##############')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n######') 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:-

```
====== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Vertical.py ======
################################
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): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 1080
Horizontal Data Set (Columns): 3
################################
>>>
```

C. Island Style.

Code:

```
import sys
import os
import pandas
as pd
import sqlite3 as sq
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, conn1)
############################################################# ########
print('###################################') sTable =
'Dim-BMI'
print('Loading :',sDatabaseName,'Table:',sTable)

sSQL="SELECT\
   Height,\
   Weight,\
   Indicator\
      FROM[Dim-BMI]\
      WHERE Indicator>2\
      ORDER BY \
         Height,\
         Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#############################################################
########
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)
print('###################################')
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:-

```
======= RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Island.py =======
################################
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): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 771
Horizontal Data Set (Columns): 3
################################
>>>
```

D. Secure Vault Style.

Code:

```
import sys
import os
import pandas
as pd
import sqlite3 as sq
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, conn1)
############################################################## ########
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, conn1)
##############################################################
########
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:-

```
==== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Secure-Vault.py ====
################################
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
################################
```

F. Association Rule Mining.

Code:

```
import sys
import os
import pandas
as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules Base='C:/VKHCG'
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_threshol d=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_colna mes=True)
    rules2=association_rules(frequent_itemsets2,metric="lift",min_thresh old=1)

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

Output:-

```
== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Association-Rule.py ==
################################
Working Base : C:/VKHCG  using  win32
################################
(541909, 8)
                  antecedents  ... conviction
0   (ALARM CLOCK BAKELIKE PINK)  ...   3.283859
1  (ALARM CLOCK BAKELIKE GREEN)  ...   3.791383
2  (ALARM CLOCK BAKELIKE GREEN)  ...   4.916181
3    (ALARM CLOCK BAKELIKE RED)  ...   5.568878
4   (ALARM CLOCK BAKELIKE PINK)  ...   3.293135

[5 rows x 9 columns]
ALARM CLOCK BAKELIKE GREEN
340.0
ALARM CLOCK BAKELIKE RED
316.0
                       antecedents  ... conviction
0   (PLASTERS IN TIN CIRCUS PARADE)  ...   2.076984
7        (PLASTERS IN TIN SPACEBOY)  ...   2.011670
11    (RED RETROSPOT CHARLOTTE BAG)  ...   5.587746

[3 rows x 9 columns]
### Done!! ############################################
>>>
```

PRACTICAL 9

Aim: Generating Reports.

A. Report Superstep.

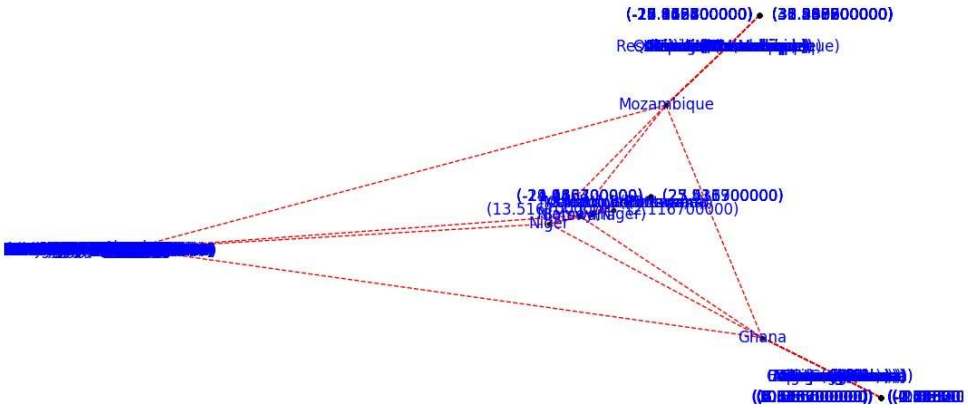
Code:-

```
#########################################################
######## 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-NetworkRouting-
Customer.gml'
sOutputFileName2='06-Report/01-EDS/02-Python/Report-NetworkRouting-
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() for i in
range(CustomerData.shape[0]):    for j in
range(CustomerData.shape[0]):
      Node0=CustomerData['Customer_Country_Name'][i]
Node1=CustomerData['Customer_Country_Name'][j]        if Node0
!= Node1:
         G.add_edge(Node0,Node1)

for i in range(CustomerData.shape[0]):
   Node0=CustomerData['Customer_Country_Name'][i]
   Node1=CustomerData['Customer_Place_Name'][i] + '('+
CustomerData['Customer_Country_Name'][i] + ')'
   Node2='('+ "{:.9f}".format(CustomerData['Customer_Latitude'][i])
+ ')\
   ('+ "{:.9f}".format(CustomerData['Customer_Longitude'][i]) + ')'    if Node0
!= Node1:
      G.add_edge(Node0,Node1)
if Node1 != Node2:
      G.add_edge(Node1,Node2)

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='sansserif',font_
color='b') plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
############################################################
######## print('###################################')
print('### Done!! ######################')
print('###################################')
############################################################
########
```

Output:-



B. Picking Content for Billboards.

Code:

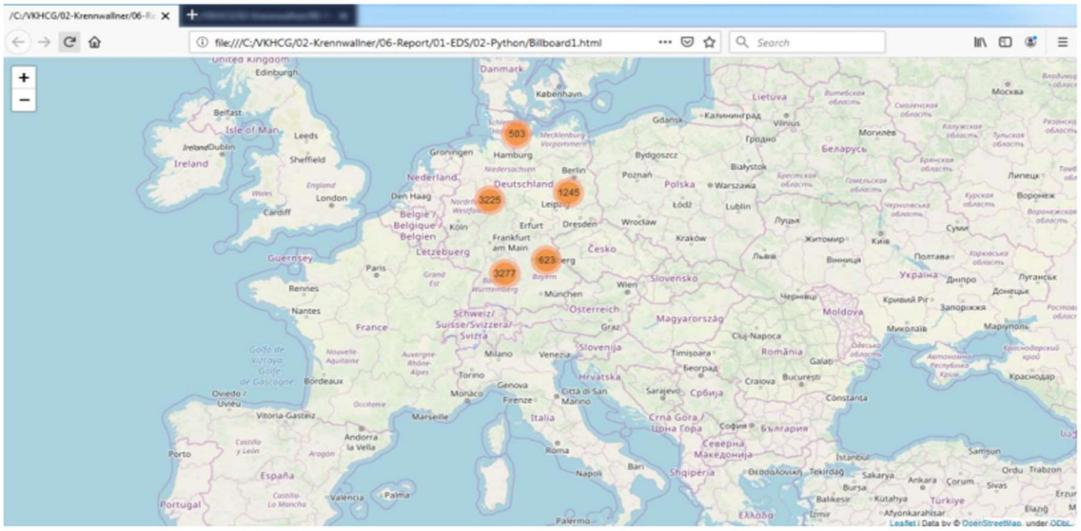
```
############################################################
# -*- coding: utf-8 -*-
############################################################
```

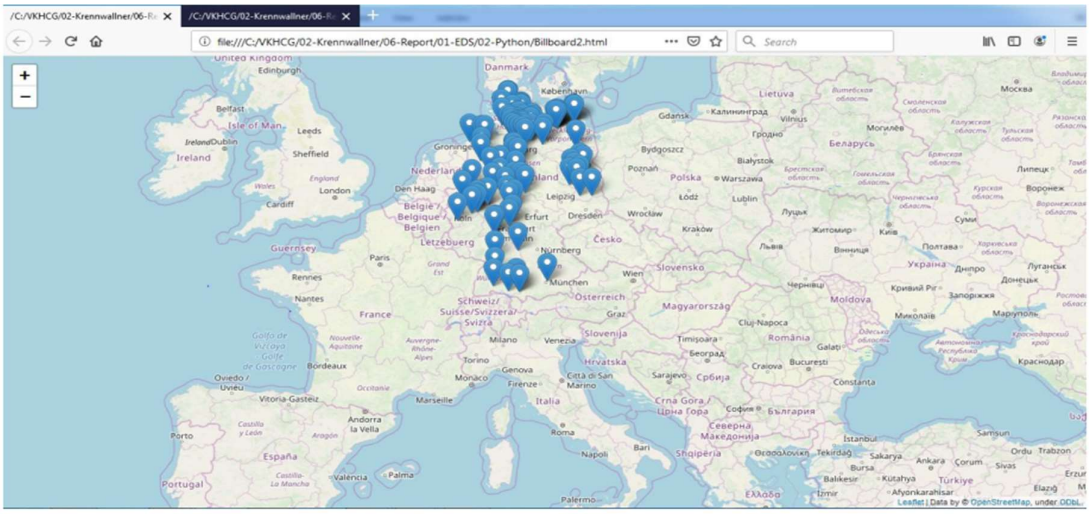
```
########
import sys
import os
import pandas
as pd
from folium.plugins import FastMarkerCluster, HeatMap from
folium import Marker, Map import webbrowser
#############################################################
########
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG' else:
   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)

t
r
y
:
      sLatitude=df["Latitude"][i]
sLatitude=float(sLatitude)     except
Exception:
      sLatitude=float(0.0)

t
r
y
:
      sDescription=df["Place_Name"][i] + ' (' + df["Country"][i]+')'     except
Exception:
      sDescription='VKHCG'
```

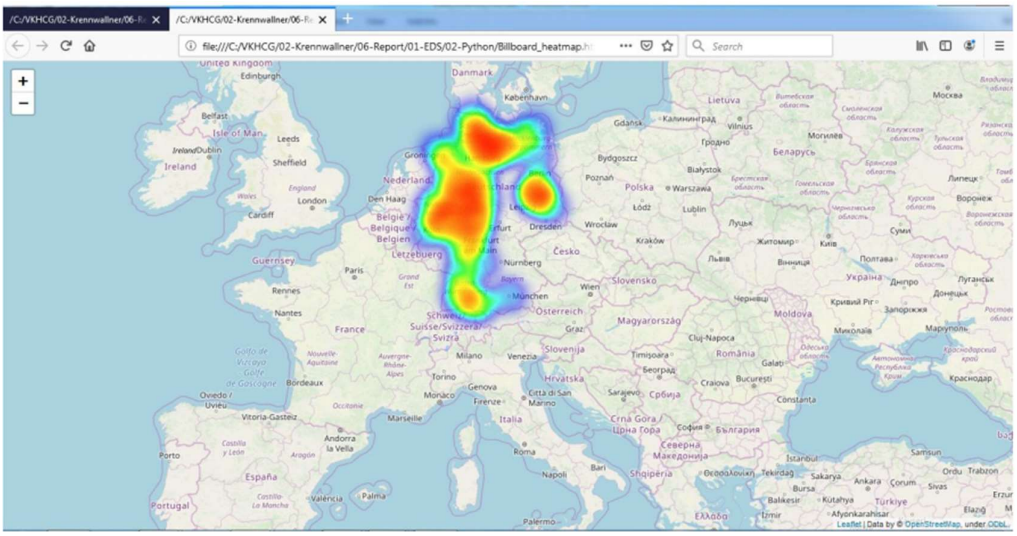
```
   if sLongitude != 0.0 and sLatitude != 0.0:         DataClusterList=list([sLatitude,
sLongitude])
      DataPointList=list([sLatitude, sLongitude, sDescription])
t+=1         if t==1:
          DataCluster=[DataClusterList]
DataPoint=[DataPointList]         else:
          DataCluster.append(DataClusterList)            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/02Python/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/02Python/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/02Python/Billboard_heatmap.html'
stops_heatmap.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
############################################################
########
print('###
Done!!
##############################################')
```

Output:-

```
PS C:\Users\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/User
s/asus/Desktop/Rinki mscit/VKHCG/01-Vermeulen/06-Report/Report Billboard.py
##################################
Working Base : C:/VKHCG  using  win32
##################################
Row: 1 of 1080.0
```







C. Reading the Containers.

Code:

```
from time import time import
numpy as np import
matplotlib.pyplot as plt from
matplotlib import offsetbox
from sklearn import (manifold, datasets, decomposition, ensemble,
discriminant_analysis, random_projection) digits =
datasets.load_digits(n_class=6) X = digits.data y = digits.target
n_samples, n_features = X.shape n_neighbors =
30 def plot_embedding(X, title=None):  x_min,
x_max = np.min(X, 0), np.max(X, 0)  X = (X -
x_min) / (x_max - x_min)
plt.figure(figsize=(10, 10))  ax =
plt.subplot(111)  for i in range(X.shape[0]):
plt.text(X[i, 0], X[i, 1], str(digits.target[i]),
color=plt.cm.Set1(y[i] / 10.),      fontdict={'weight': 'bold', 'size':
9})  if hasattr(offsetbox, 'AnnotationBbox'):  # only print
thumbnails with matplotlib > 1.0      shown_images = np.array([[1.,
1.]]) # just something big  for i in range(digits.data.shape[0]):
dist = np.sum((X[i] - shown_images) ** 2, 1)      if np.min(dist) <
4e-3:
# don't show points that are too close
continue
shown_images = np.r_[shown_images, [X[i]]]        imagebox =
offsetbox.AnnotationBbox(offsetbox.OffsetImage(digits.images[i],
cmap=plt.cm.gray_r),
X[i])
ax.add_artist(imagebox)
plt.xticks([]), plt.yticks([])
```

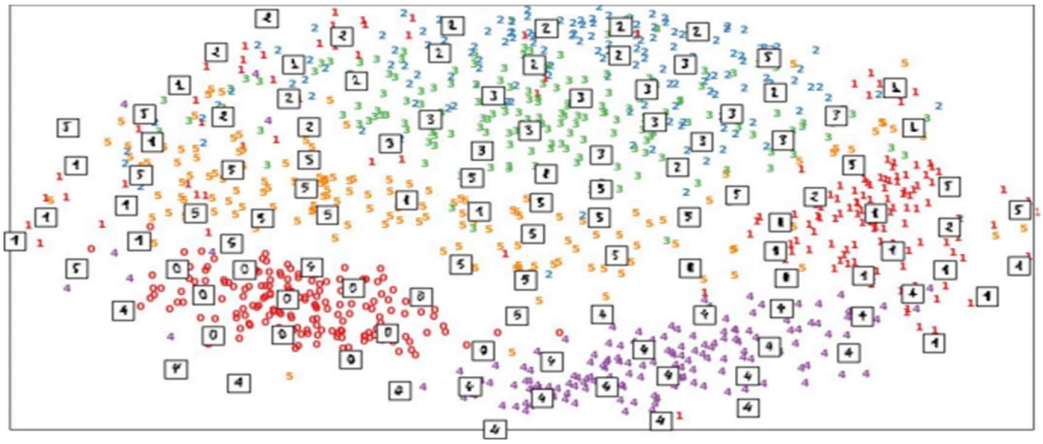
```
if tite is not none:
plt.title(title)
n_img_per_row = 20        img = np.zeros((10 * n_img_per_row, 10 *
n_img_per_row))        for i in range(n_img_per_row):            ix = 10 * i +
1            for j in range(n_img_per_row):
iy = 10 * j + 1
img[ix:ix + 8, iy:iy + 8] = X[i * n_img_per_row + j].reshape((8, 8))
plt.figure(figsize=(10, 10))
plt.imshow(img, cmap=plt.cm.binary)
plt.xticks([])                plt.yticks([])
plt.title('A selection from the 64-dimensional digits dataset')                print("Computing
random projection")
rp =
random_projection.SparseRandomProjection(n_components=2, random_state=42)
X_projected = rp.fit_transform(X)
plot_embedding(X_projected, "Random Projection of the digits")
print("Computing PCA projection")                t0
= time()                X_pca =
decomposition.TruncatedSVD(n_components=2).fit_transform(X)
plot_embedding(X_pca,"Principal Components projection of the digits (time
%.2fs)" %
(time() - t0))
print("Computing Linear Discriminant Analysis projection")
X2 = X.copy()
X2.flat[::X.shape[1] + 1] += 0.01 # Make X invertible                t0 =
time()                X_lda =
discriminant_analysis.LinearDiscriminantAnalysis(n_components=2)
fit_transform(X2, y)                plot_embedding(X_lda,
"Linear Discriminant projection of the digits (time %.2fs)"
%(time() - t0))
print("Computing Isomap embedding")                t0 =
time()
X_iso = manifold.Isomap(n_neighbors,
n_components=2).fit_transform(X)
print("Done.")
plot_embedding(X_iso,
"Isomap projection of the digits (time %.2fs)" %(time() - t0))
print("Computing LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,
method='standard')
t0 = time()
X_lle = clf.fit_transform(X)                print("Done.
Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_lle,
"Locally Linear Embedding of the digits (time %.2fs)"
```

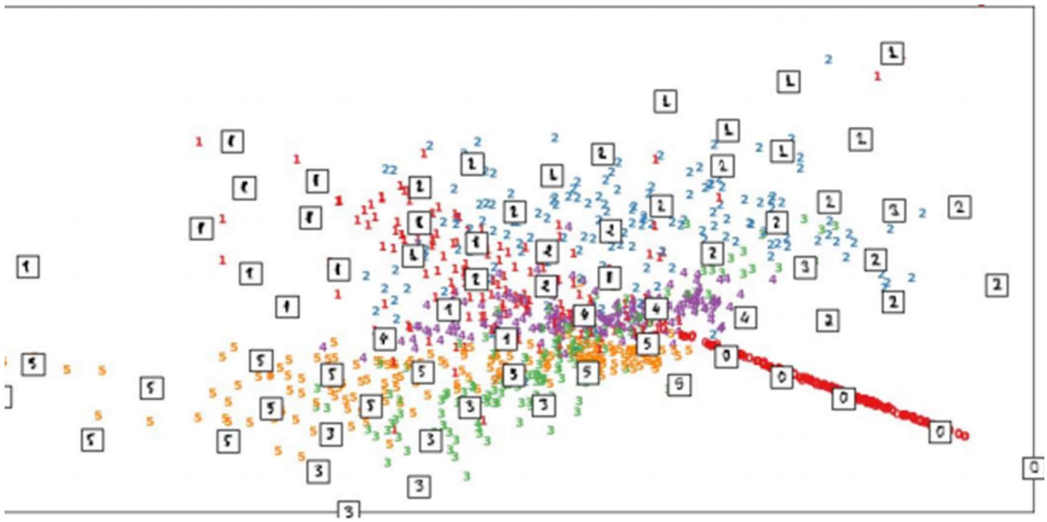
```
%(time() - t0))                print("Computing modified LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,
method='modified')
t0 = time()
X_mlle = clf.fit_transform(X)                print("Done.
Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_mlle,"Modified Locally Linear Embedding of the
digits (time %.2fs)" %(time() - t0))                print("Computing Hessian
LLE embedding")                clf =
manifold.LocallyLinearEmbedding(n_neighbors,
n_components=2,method='hessian')
t0 = time()
X_hlle = clf.fit_transform(X)                print("Done.
Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_hlle,"Hessian Locally Linear Embedding of
the digits (time %.2fs)" %(time() - t0))
print("Computing LTSA embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors,
n_components=2,method='ltsa')
t0 = time()
X_ltsa = clf.fit_transform(X)                print("Done.
Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_ltsa,"Local Tangent Space Alignment
of the digits (time %.2fs)" %
(time() - t0))
print("Computing MDS embedding")                clf =
manifold.MDS(n_components=2, n_init=1, max_iter=100)
t0 = time()
X_mds = clf.fit_transform(X)                print("Done.
Stress: %f" % clf.stress_)
plot_embedding(X_mds,"MDS embedding of the digits
(time %.2fs)" %
(time() - t0))
print("Computing Totally Random Trees embedding") hasher =
ensemble.RandomTreesEmbedding(n_estimators=200, random_state=0,
max_depth
=5) t0 =
time()
X_transformed = hasher.fit_transform(X)
pca = decomposition.TruncatedSVD(n_components=2) X_reduced =
pca.fit_transform(X_transformed) plot_embedding(X_reduced,
"Random forest embedding of the digits (time %.2fs)" %
(time() - t0))

print("Computing Spectral embedding")
```

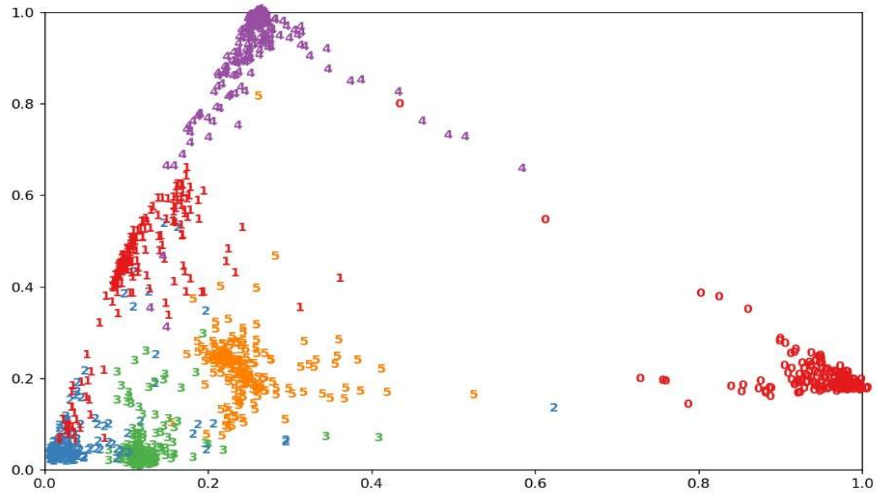
```
embedder = manifold.SpectralEmbedding(n_components=2,
random_state=0,  eigen_solver="arpack")
t0 = time()
X_se = embedder.fit_transform(X) plot_embedding(X_se,
"Spectral embedding of the digits (time %.2fs)" %
(time() - t0))
print("Computing t-SNE embedding")
tsne = manifold.TSNE(n_components=2, init='pca', random_state=0) t0 = time()
X_tsne = tsne.fit_transform(X) plot_embedding(X_tsne,
"t-SNE embedding of the digits (time %.2fs)" %
(time() - t0))
plt.show()
```

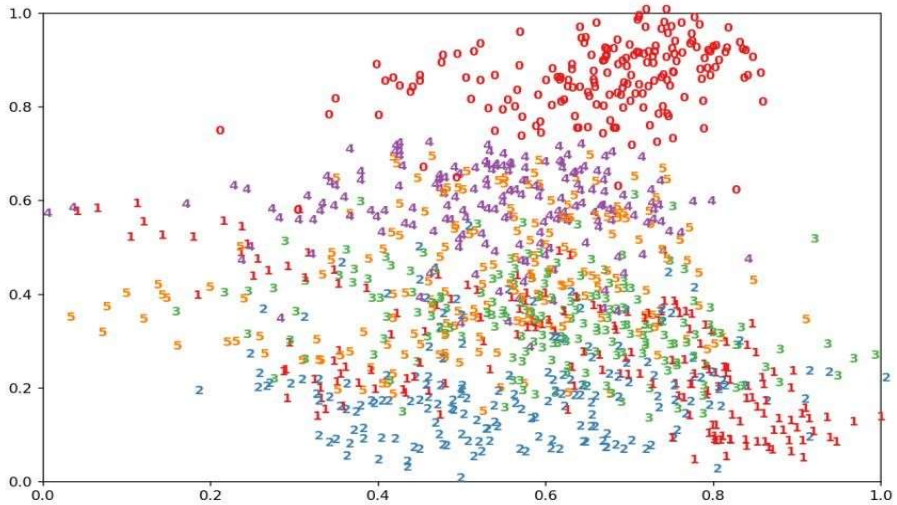
Output:-

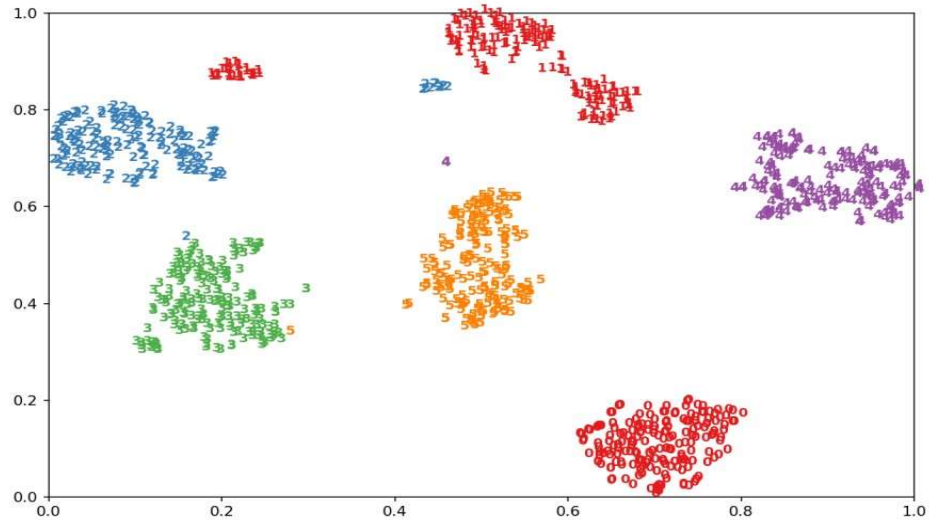
```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/VKHCG/03-Hillman/06-Report/Report_Reading_Container.py =====
1. Computing random projection
2. Computing 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
```






0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0

D.Financials

Code:

```
# -*- coding: utf-8 -*-
############################################################
########
import sys
import os
import pandas
as pd import
sqlite3 as sq
import re
from openpyxl import load_workbook
############################################################
########
Base='C:/VKHCG'
############################################################
######## print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
############################################################
######## sInputTemplateName='00-RawData/Balance-Sheet-Template.xlsx'
############################################################
########
sOutputFileName='06-Report/01-EDS/02-Python/Report-BalanceSheet'
Company='04-Clark'
############################################################
########
sDatabaseName=Base + '/' + Company + '/06-Report/SQLite/clark.db' conn =
sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
############################################################
########
### Import Balance Sheet Data
############################################################
########
for y in range(1,13):     sInputFileName='00-RawData/BalanceSheets' +
str(y).zfill(2) +
'.csv'
   sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')     print('Loading
:',sFileName)
   print('##################################')
```

```
ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=Fals e,
encoding="latin-1")
   print('##################################')

############################################################
########
   ForexDataRaw.index.names = ['RowID']
   sTable='BalanceSheets'     print('Storing
:',sDatabaseName,' Table:',sTable)     if y == 1:
print('Load Data')
       ForexDataRaw.to_sql(sTable, conn, if_exists="replace")     else:
       print('Append Data')
       ForexDataRaw.to_sql(sTable, conn, if_exists="append")
############################################################
########
sSQL="SELECT \
          Year, \
          Quarter, \
          Country, \
          Company, \
          CAST(Year AS INT) || 'Q' || CAST(Quarter AS INT) AS sDate, \
          Company || ' (' || Country || ')' AS sCompanyName , \            CAST(Year AS
INT) || 'Q' || CAST(Quarter AS INT) || '-' ||\
          Company || '-' || Country AS sCompanyFile \
       FROM BalanceSheets \
       GROUP BY \
          Year, \
          Quarter, \
          Country, \
          Company \
       HAVING Year is not null \
      ;"
sSQL=re.sub("\s\s+", " ", sSQL)
sDatesRaw=pd.read_sql_query(sSQL, conn)
print(sDatesRaw.shape) sDates=sDatesRaw.head(5)
############################################################
########
## Loop Dates
############################################################
########
for i in range(sDates.shape[0]):     sFileName=Base + '/' + Company + '/' +
sInputTemplateName     wb = load_workbook(sFileName)
ws=wb.get_sheet_by_name("Balance-Sheet")
   sYear=sDates['sDate'][i]     sCompany=sDates['sCompanyName'][i]
sCompanyFile=sDates['sCompanyFile'][i]
```

```
    sCompanyFile=re.sub("\s+", "", sCompanyFile)

    ws['D3'] = sYear
    ws['D5'] = sCompany

    sFields = pd.DataFrame(
          [
         ['Cash','D16', 1],
         ['Accounts_Receivable','D17', 1],
         ['Doubtful_Accounts','D18', 1],
         ['Inventory','D19', 1],
         ['Temporary_Investment','D20', 1],
         ['Prepaid_Expenses','D21', 1],
         ['Long_Term_Investments','D24', 1],
         ['Land','D25', 1],
         ['Buildings','D26', 1],
         ['Depreciation_Buildings','D27', -1],
         ['Plant_Equipment','D28', 1],
         ['Depreciation_Plant_Equipment','D29', -1],
         ['Furniture_Fixtures','D30', 1],
         ['Depreciation_Furniture_Fixtures','D31', -1],
         ['Accounts_Payable','H16', 1],
         ['Short_Term_Notes','H17', 1],
         ['Current_Long_Term_Notes','H18', 1],
         ['Interest_Payable','H19', 1],
         ['Taxes_Payable','H20', 1],
         ['Accrued_Payroll','H21', 1],
         ['Mortgage','H24', 1],
         ['Other_Long_Term_Liabilities','H25', 1],
         ['Capital_Stock','H30', 1]
         ]
          )

    nYear=str(int(sDates['Year'][i]))     nQuarter=str(int(sDates['Quarter'][i]))
sCountry=str(sDates['Country'][i])
    sCompany=str(sDates['Company'][i])

    sFileName=Base + '/' + Company + '/' + sOutputFileName + \
    '-' + sCompanyFile + '.xlsx'

    print(sFileName)

    for j in range(sFields.shape[0]):

       sSumField=sFields[0][j]         sCellField=sFields[1][j]
```

```
        nSumSign=sFields[2][j]

        sSQL="SELECT  \
              Year, \
              Quarter, \
              Country, \
              Company, \
              SUM(" + sSumField + ") AS nSumTotal \
            FROM BalanceSheets \
            GROUP BY \
              Year, \
              Quarter, \
              Country, \
              Company \
            HAVING \
               Year=" + nYear + " \
            AND \
               Quarter=" + nQuarter + " \
            AND \
               Country='" + sCountry + "' \
            AND \
               Company='" + sCompany + "' \
           ;"
        sSQL=re.sub("\s\s+", " ", sSQL)
sSumRaw=pd.read_sql_query(sSQL, conn)        ws[sCellField] =
sSumRaw["nSumTotal"][0] * nSumSign          print('Set cell',sCellField,'
to ', sSumField,'Total')
        wb.save(sFileName)
```

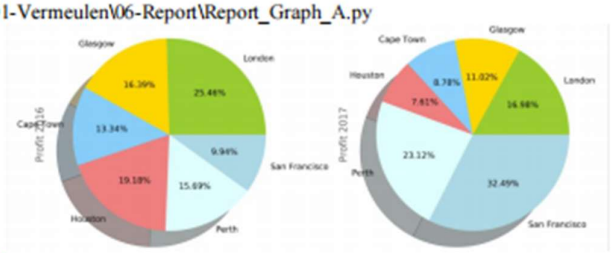
Output:-

Graphics
This section will now guide you through a number of visualizations that particularly useful in presenting data to my customers.
Pie Graph
Double Pie
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py

Pie Graph

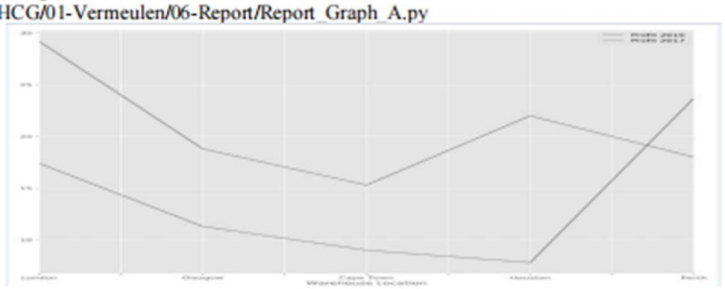
Double Pie

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py



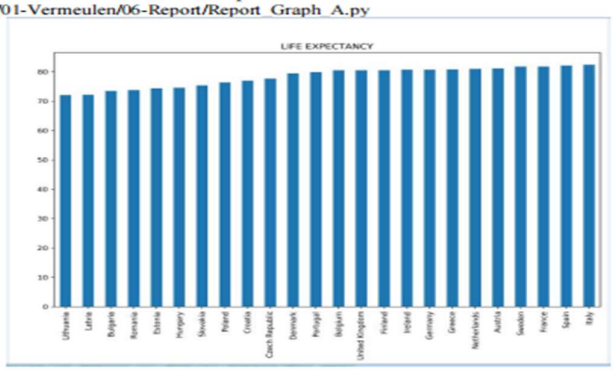
Line Graph

C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py



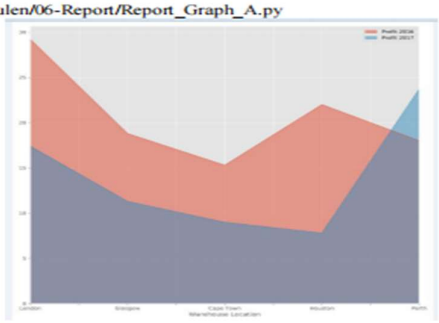
Bar Graph / Horizontal Bar Graph

C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py

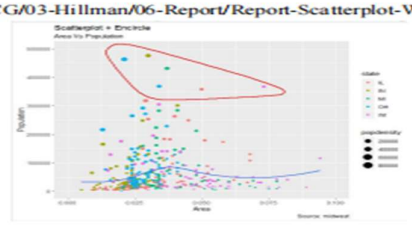


Area Graph

C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py

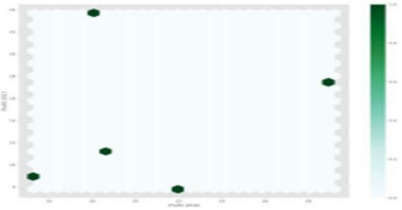


Scatter Graph : VKHCG/03-Hillman/06-Report/Report-Scatterplot-With-Encircling.r



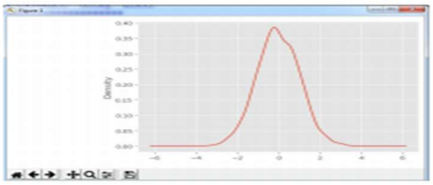
Hexbin:

Program : C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py



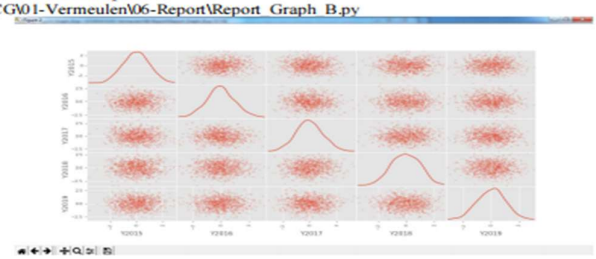
Kernel Density Estimation (KDE) Graph

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py



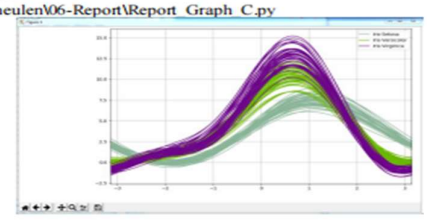
Scatter Matrix Graph

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py



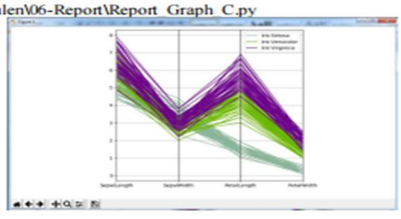
Andrews' Curves

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py

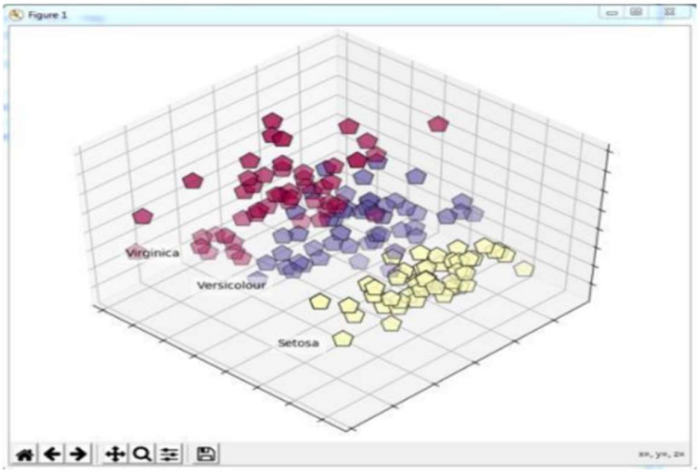


Parallel Coordinates

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py



```
PS C:\Users\asus> & C:/Users/asus/AppData/Local/Programs/Python/Python312/python.exe c:/User
s/asus/Desktop/Rinki_mscit/VKHCG/01-Vermeulen/06-Report/Report_PCA_IRIS.py
```



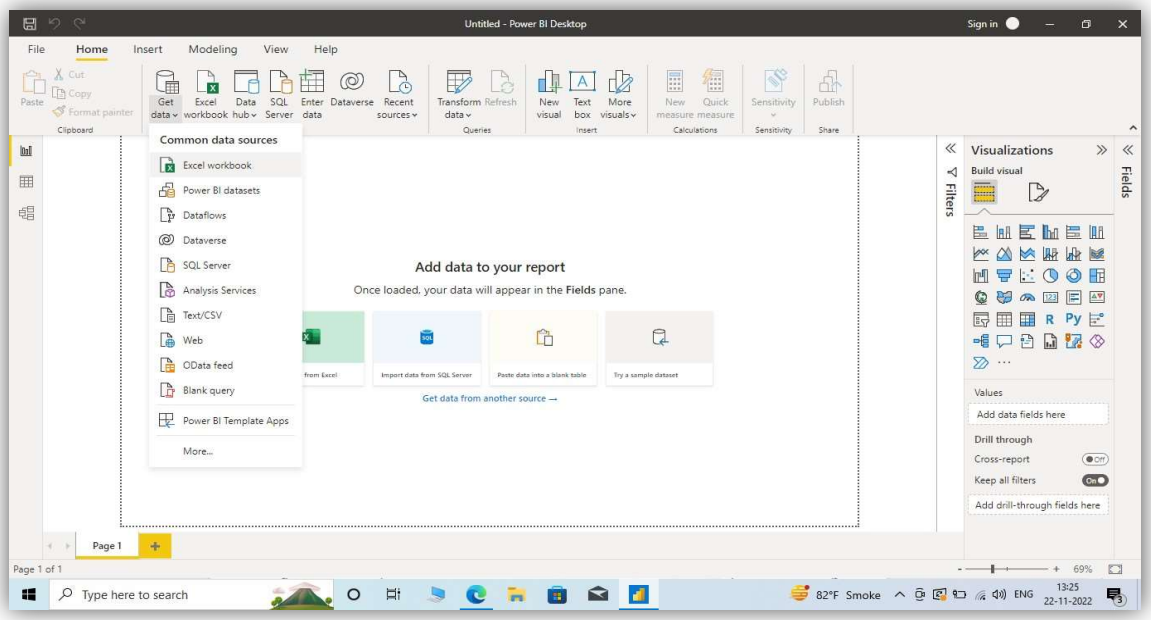
PRACTICAL 10

Aim: Data Visualization with Power Bi.

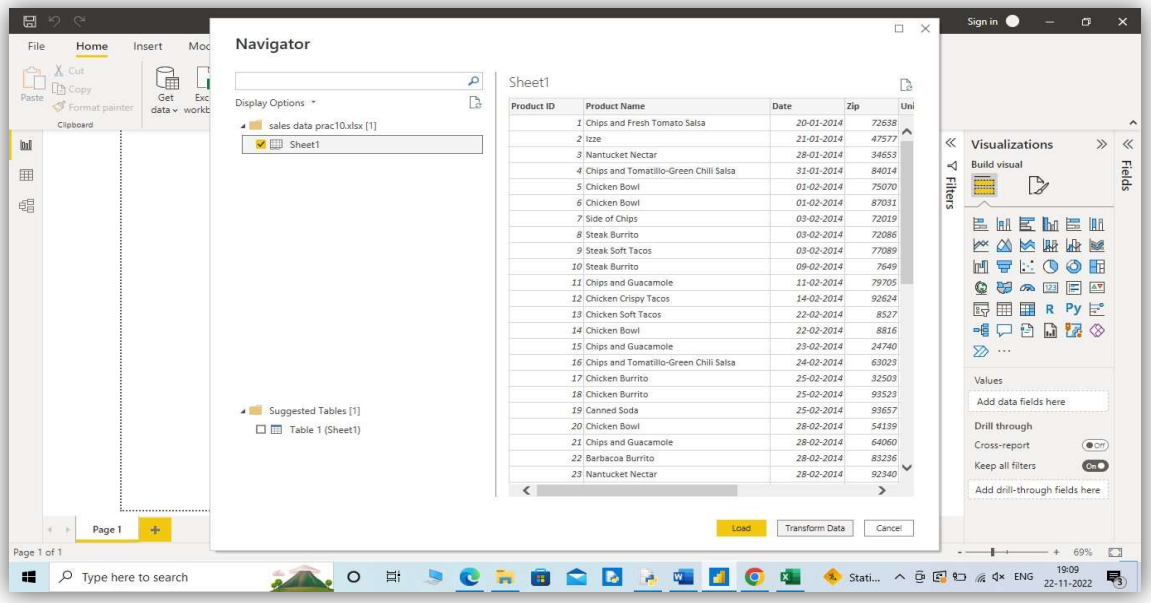
Case Study: Sales Data.

Task 1: Connect to an Excel Workbook.

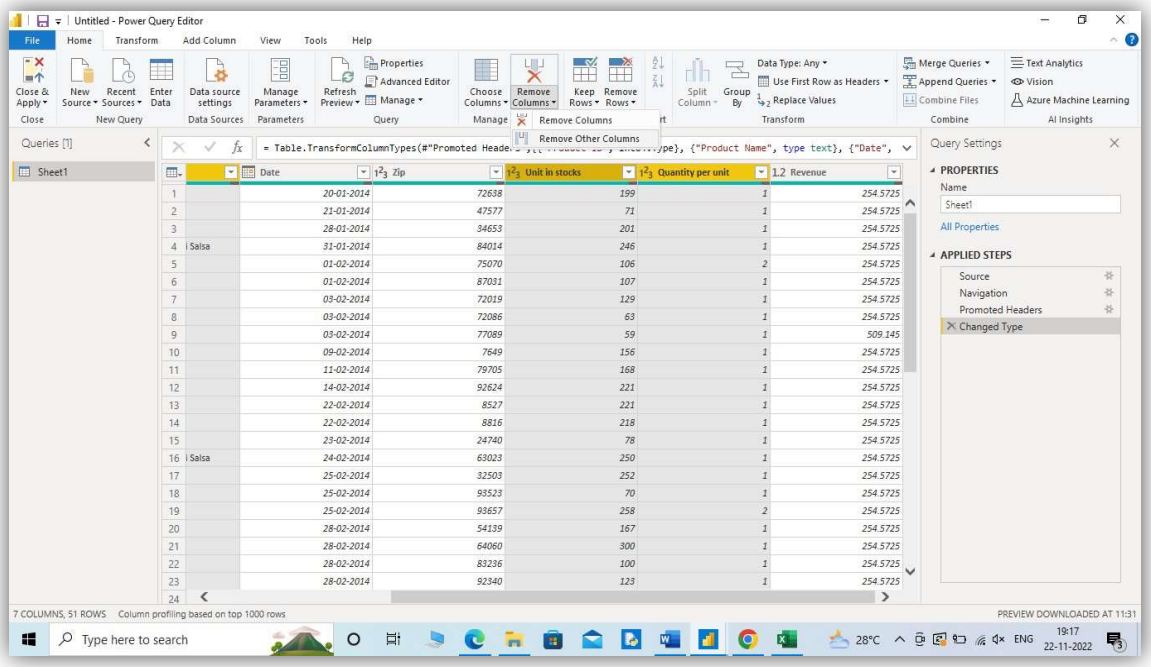
1. Launch power Bi Desktop
2. From the Home ribbon, select **Get Data** and click on Excel **or** by directly clicking on **Excel Workbook.**



3. In the Open file select dialog box, select your .xlsx file.
4. Directly load whole sheet and open **Query Editor or** click on **transform data** to directly open sheet in Query Editor **(Necessary columns:- Product ID, Product Name, Units in stock and Quantity per unit).**



5. Select all the necessary columns by **ctrl + click** and Go to Remove columns on Home Ribbon and Select **Remove other columns**



6. Change the data type of the Units In Stock column

For the excel workbook, Products in the stocks will always be a whole number, so in this step you confirm Units In Stock column's data type is Whole number

a. Select the Units In Stock column
b. Select the Data Type drop-down button in home ribbon
c. If not already a whole number, select the whole number for Data Type from the drop-down (the data type button also displays the current data type)

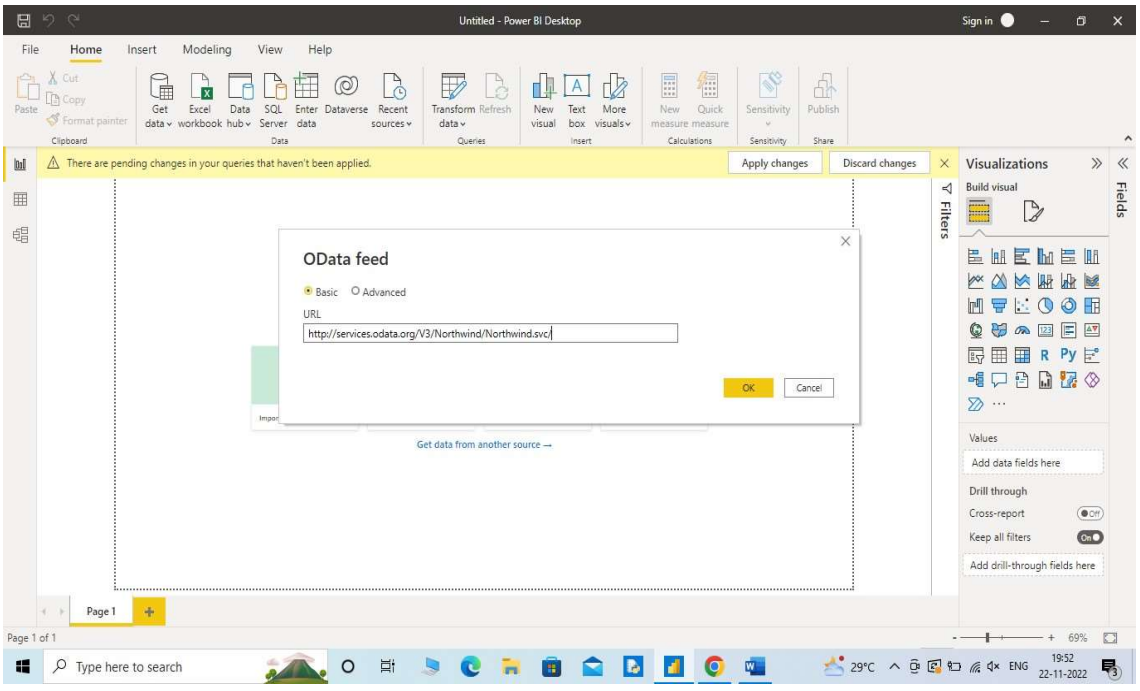
Task 2: Import order data from an OData feed

You import data into Power Bi Desktop from the link given below

http://services.odata.org/V3/Northwind/Northwind.svc/

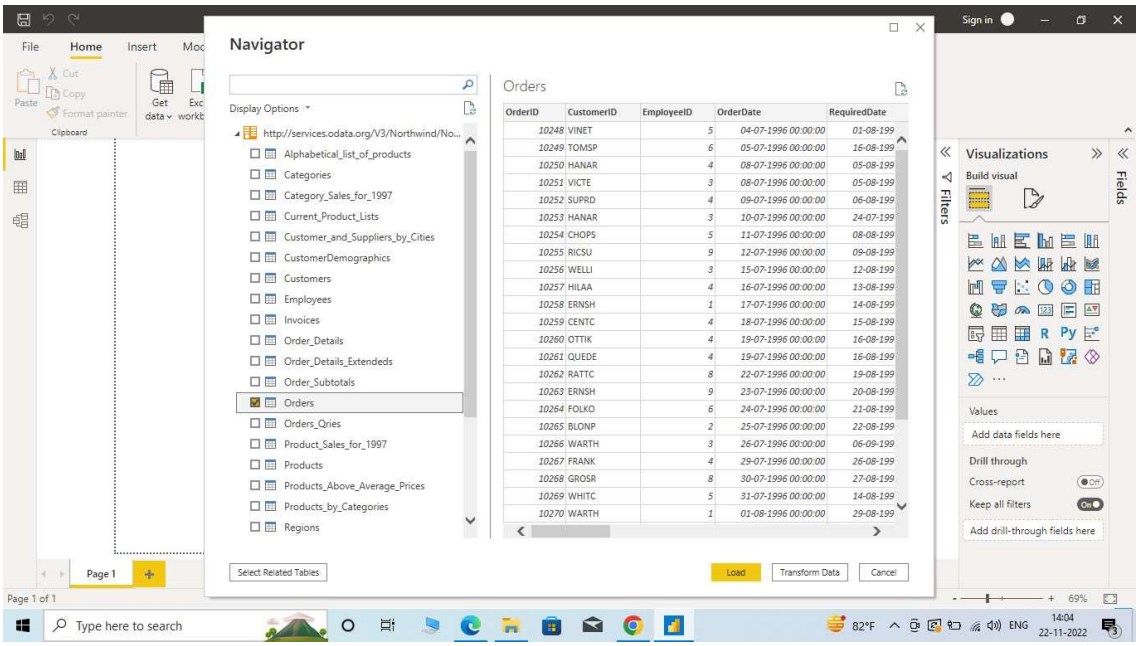
I. Connect to an OData feed

1. From the Home Ribbon tab in Query Editor, select Get data
2. Browse OData feed data source
3. In the OData Feed dialog box, type or copy the link from above 4. Select Ok.



5. Select orders column

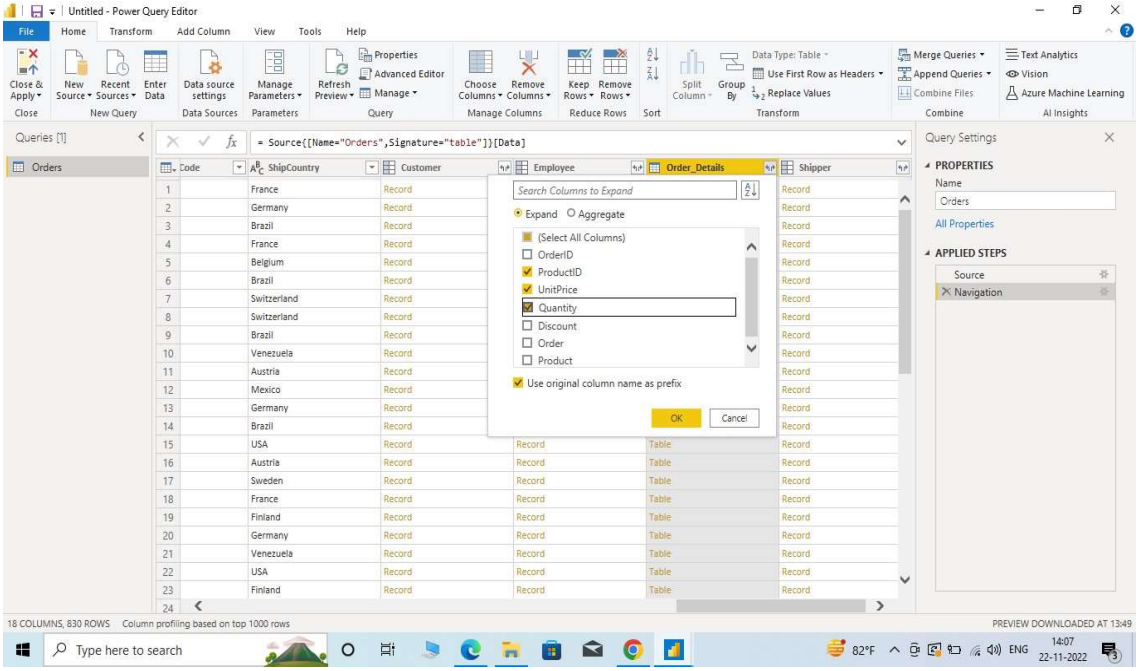
6. And load it.



II. Expand orders column

Expand the Order_Details table that is related to the orders table, to combine the ProductID, UnitPrice and Quantity columns from Order_Details into the order table.

After you expanded new columns will be added

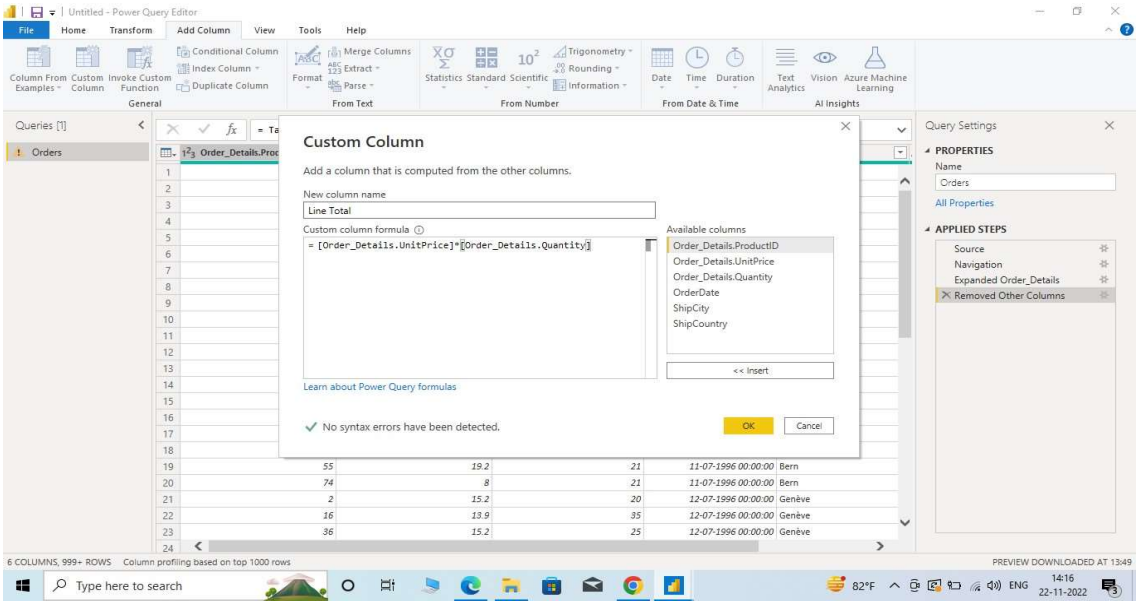


III. Removing Unwanted columns and Displaying column of interest

1. In the Query View, select all necessary column (**OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice** and **Order_Details.Quantity**)
2. Select all Necessary column by CTRL + CLICK
3. Go to remove columns on home ribbon and select **remove other columns**

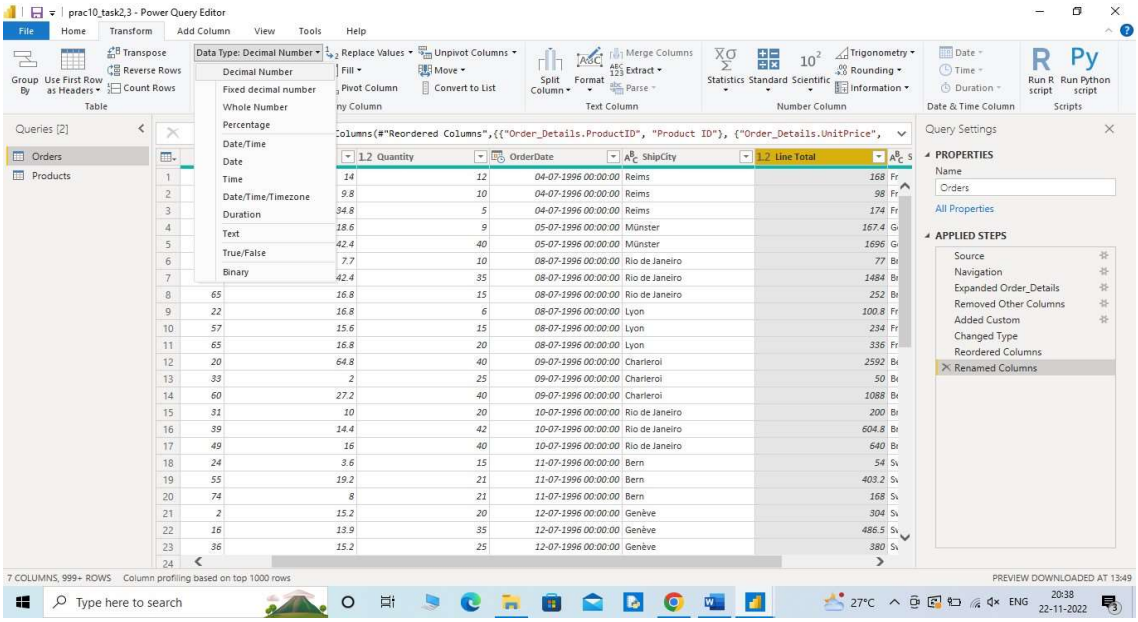
IV. Calculate the line Total for each Order_Details column

1. Go to add column from home ribbon and select add custom column
2. In the add custom column dialog box, in the custom column formula textbox, enter **[Order_Details.UnitPrice]*[Order_Details.Quantity]** .
3. In the New column name , Enter Line Total.



V. Set the data type of the Line Total field

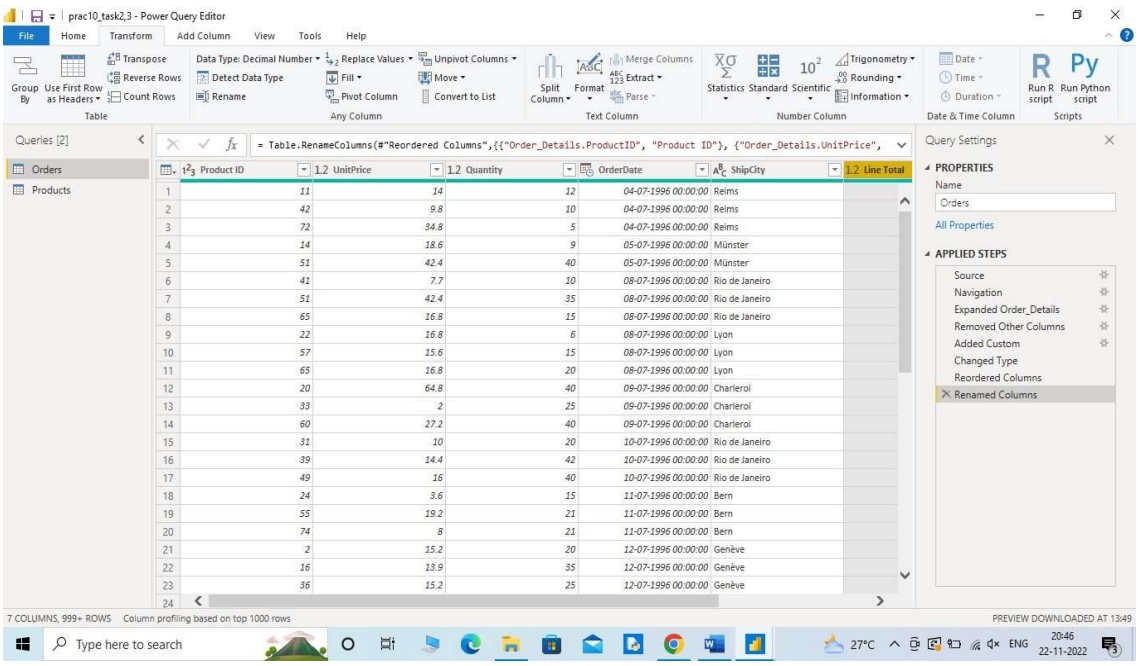
1. Right click the LineTotal column
2. In Query view, Select data type in Transform ribbon and choose Decimal number.



VI. Rename and reorder columns In query

1. In Query Editor , drag the LineTotal column to the left, after ShipCountry

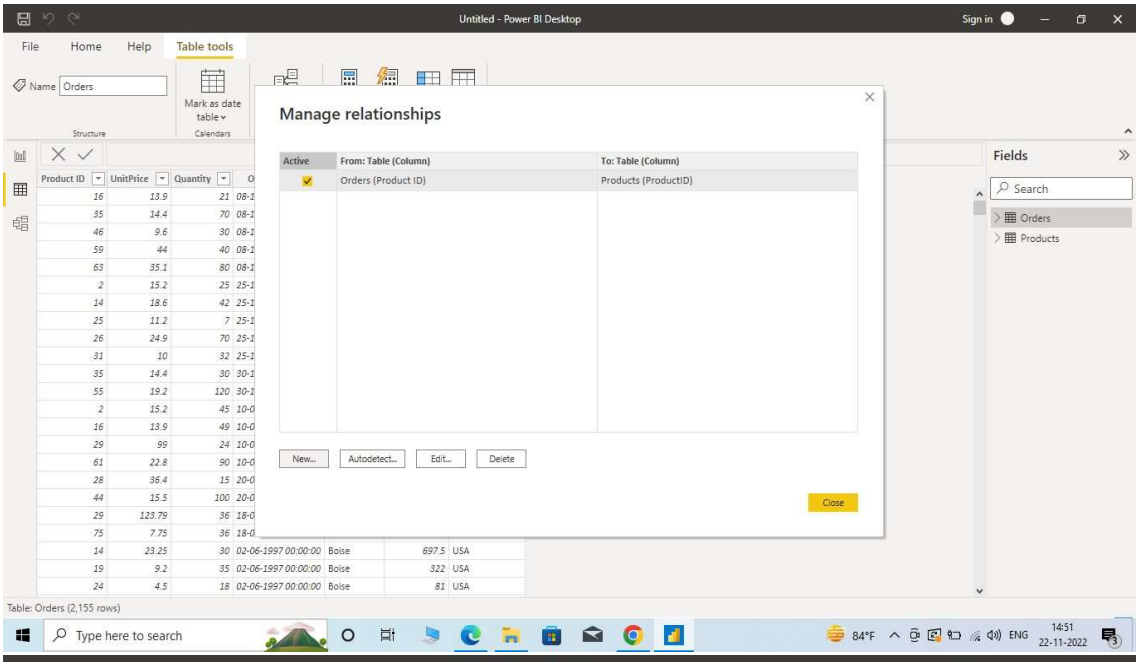
2. Remove the Order_Details prefix from **Order_Details.ProductID, Order_Details.UnitPrice** and **Order_Details.Quantity** columns, by doubling clicking in each column header

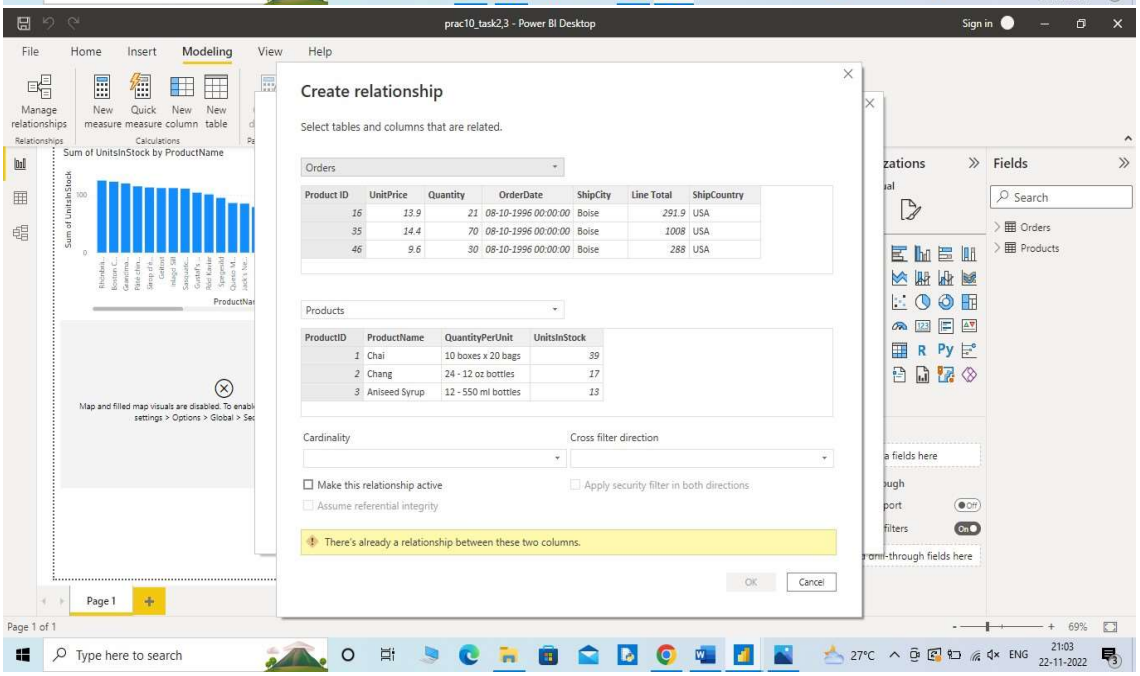


Task 3: Combine the products and Total sales queries

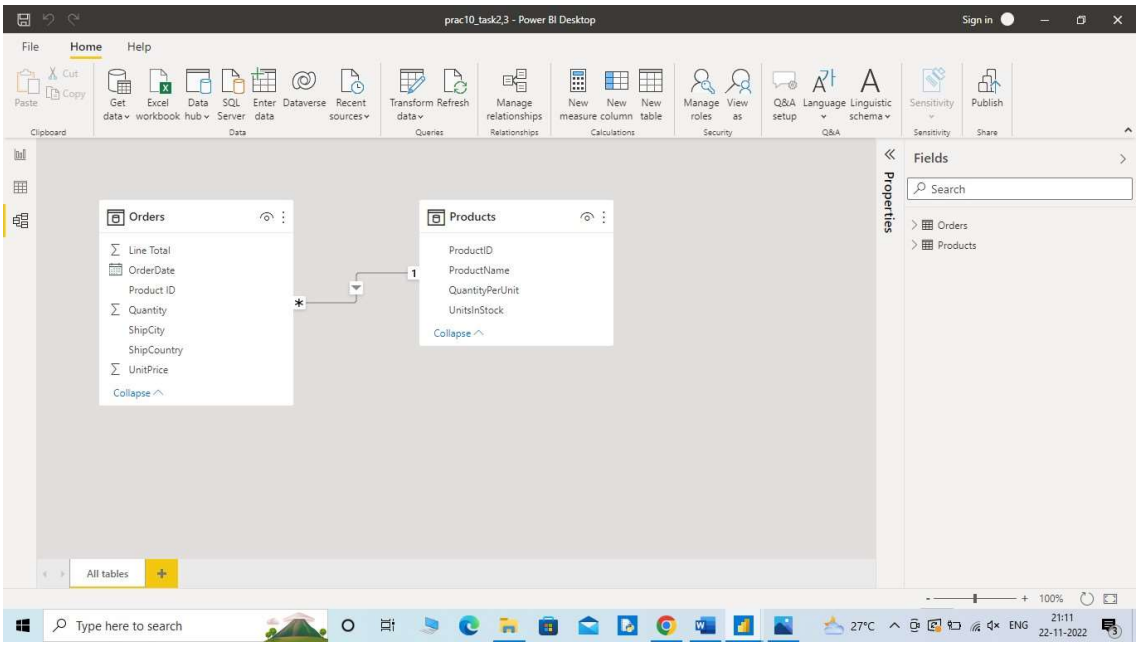
1. Add a new table on a different sheet from same OData feed (Products)
2. Confirm the relationship between products and Total Sales Load the model out of the Query Editor by top left button close and apply from home ribbon.
3. Power Bi Desktop Loads the data from the two Query
4. Once the data is loaded, select the Manage Relationship from home ribbon (Note: - Manage Relationship button will only active there are two or more separate table in same BI file)
5. Select the New…. Button

6. When we attempt to create the relationship, if we see that one already exists! As shown in picture below then cancel to get back





7. Select cancel and then select relationship view in power BI Desktop



Task 4: Build Visuals using your data

1. Create chart showing UnitsInStock by Products and Total sales by year.