SIES (NERUL) COLLEGE OF ARTS, SCIENCE AND COMMERCE

NAAC ACCREDITED 'A' GRADE COLLEGE

(ISO 9001:2015 CERTIFIED INSTITUTION)

NERUL, NAVI MUMBAI - 400706

Seat No: 3713538

Certified that VARMA VISHAL VIJAY

Of Class MSC.IT PART-1 has duly completed the practical course in the subject of DATA SCIENCE during the academic year 2021-22 as per the syllabus prescribed by the University of Mumbai.

Subject Teacher **External Examiner**

Head of Department **Principal**

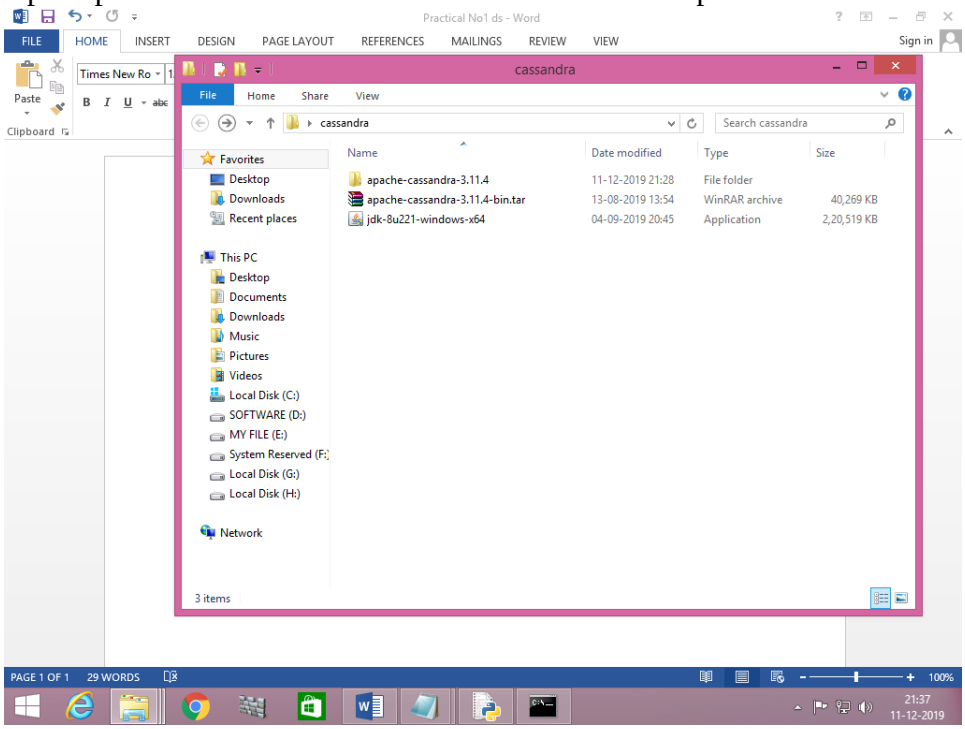
Index

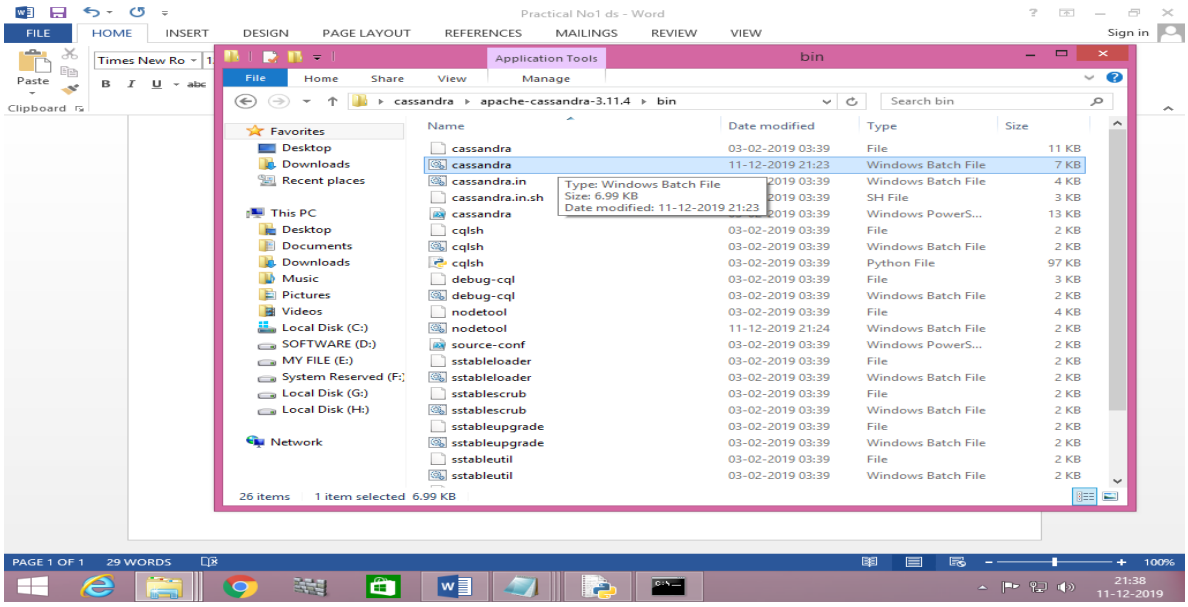
Practical No. 1

Aim: Create data model using Cassandra.

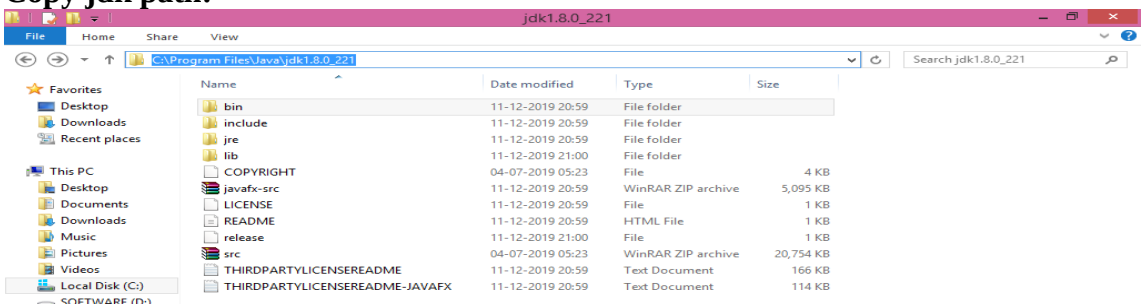
Required Software:
Java v1.8, Python v2.7, Cassandra File

Cassandra setup and configuration:
Open apache Cassandra >bin>Cassandra edit with notpad

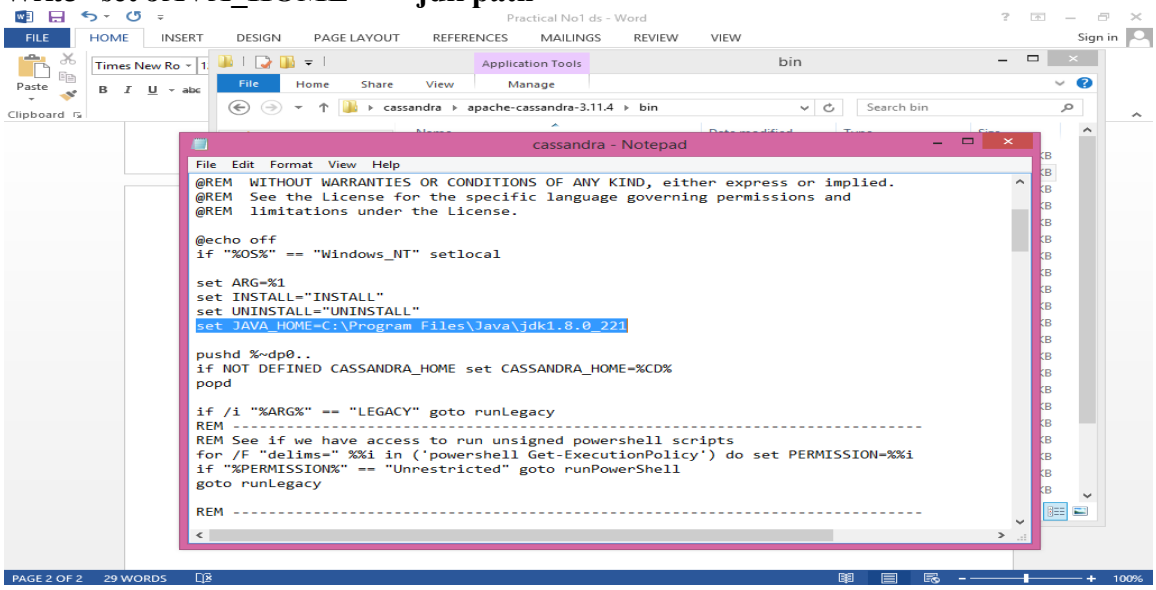




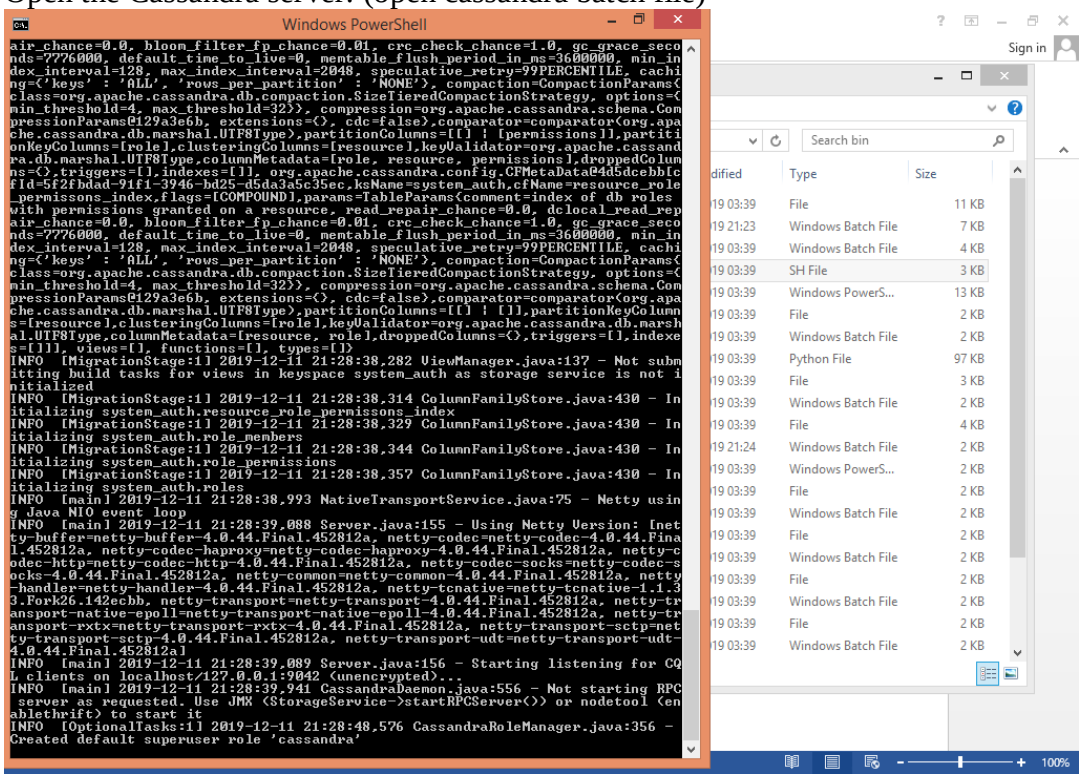
Copy jdk path:



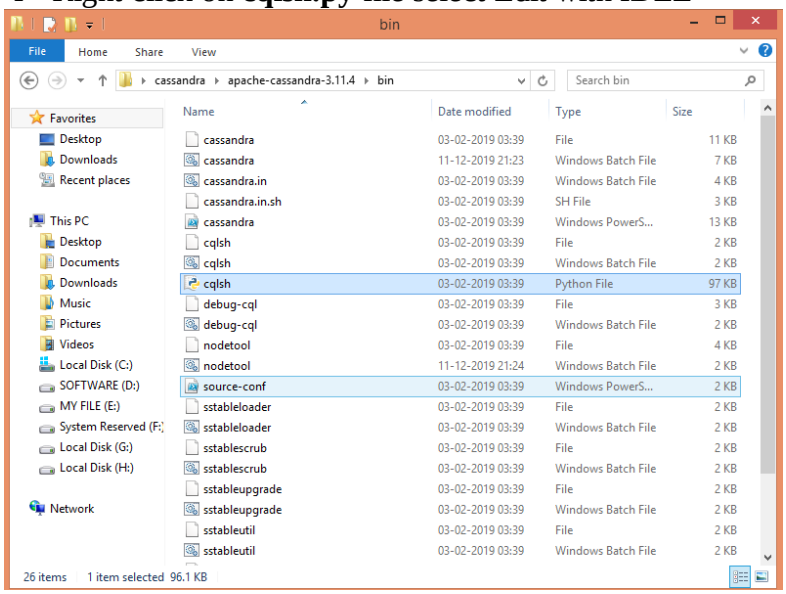
Set environmental variable in Cassandra.
Write **“set JAVA_HOME=” + jdk path**



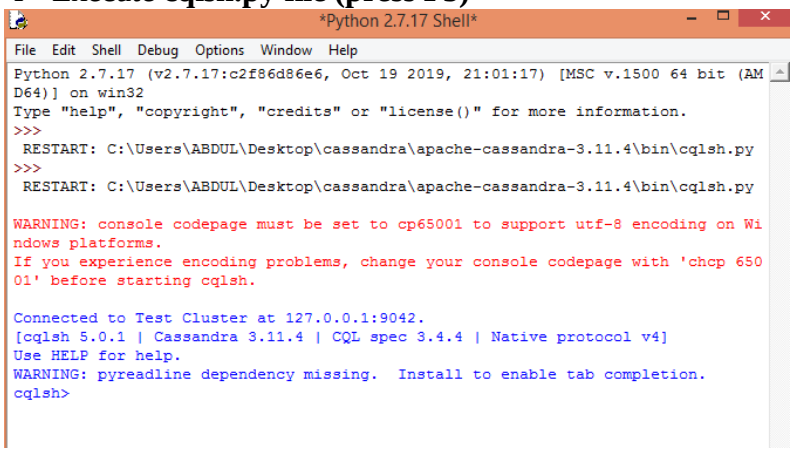
Open the Cassandra server: (open cassandra batch file)



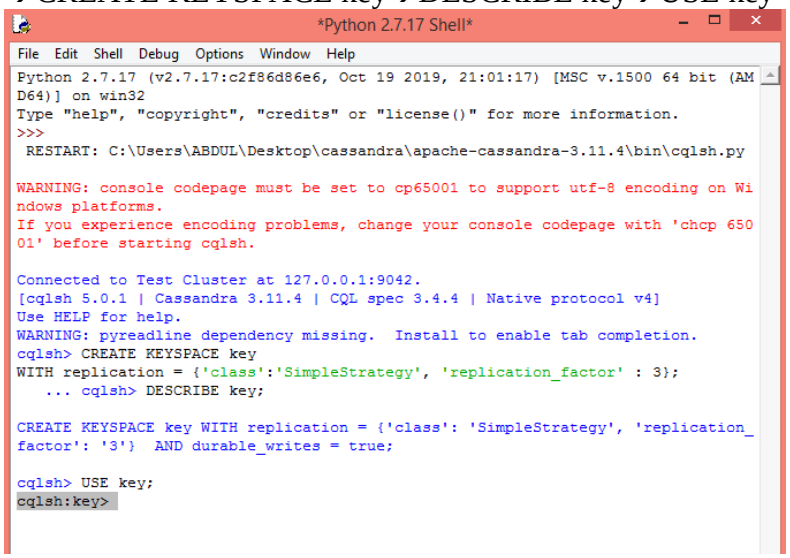
➔ Right click on **cqlsh.py** file select Edit with IDLE



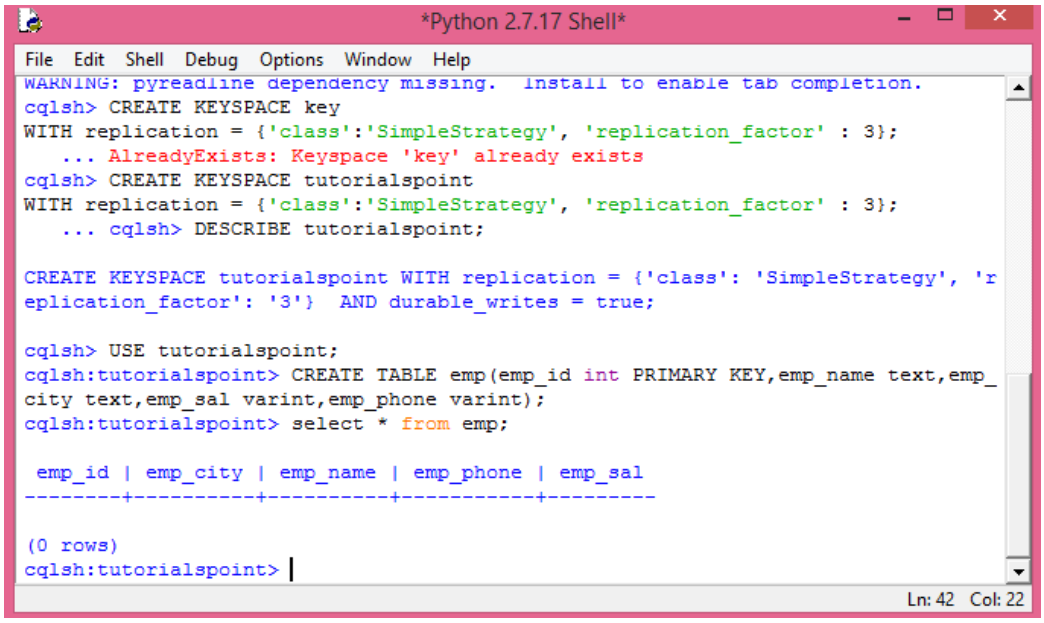
➔ Execute **cqlsh.py file (press F5)**



Creating a Keyspace using Cqlsh
→CREATE KEYSPACE key→DESCRIBE key→USE key

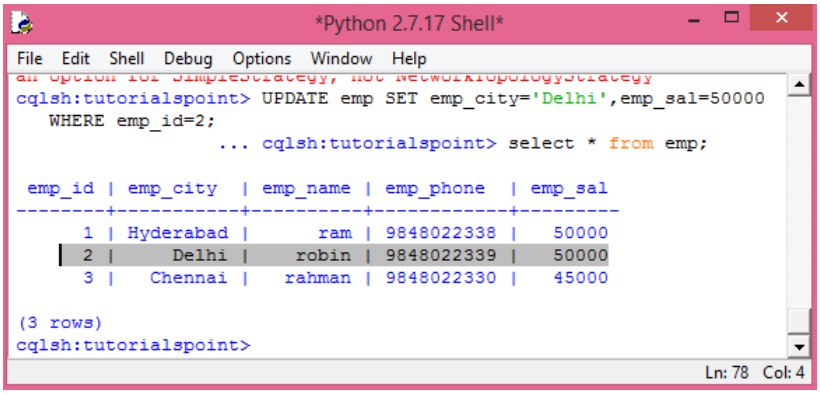


→Create table:

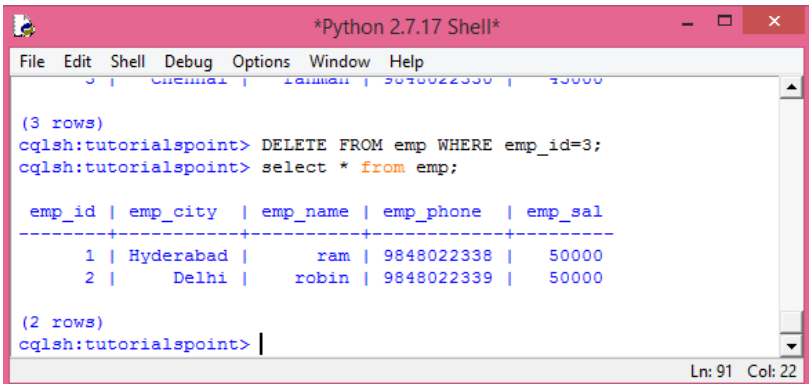


➔ Insert the data into the table of emp

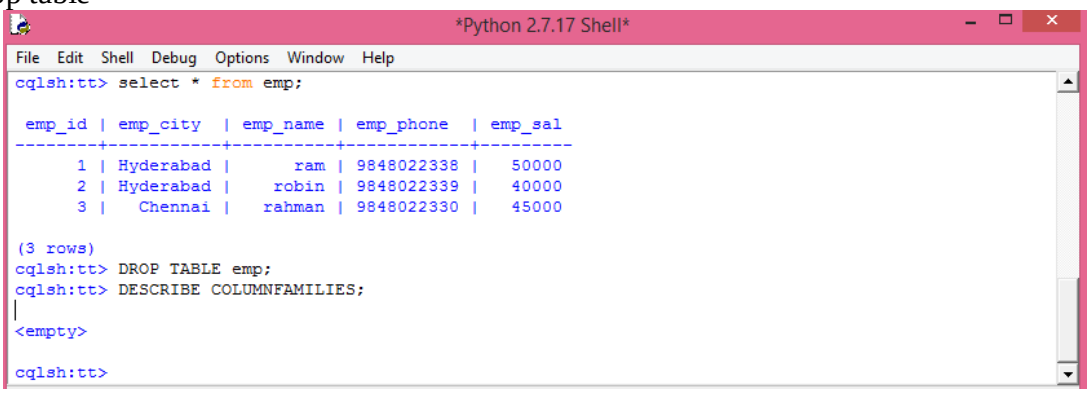
➔ Update DATA



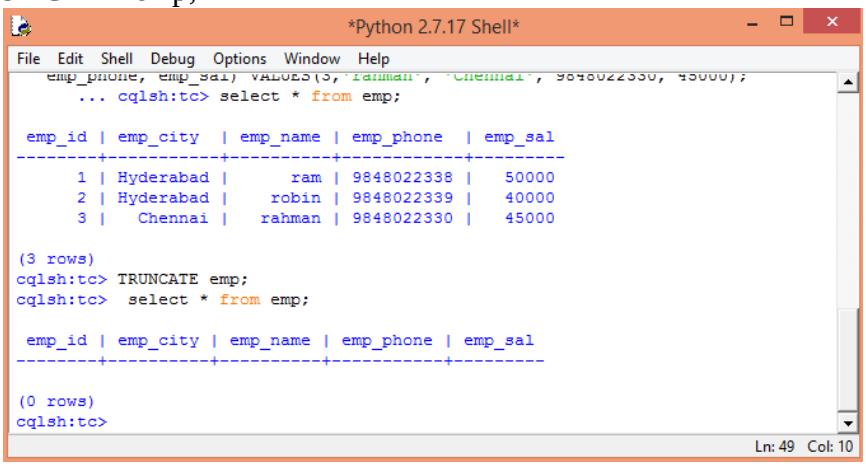
➔ Delet data



➔ Drop table



➔ TRUNCATE emp;



Practical No. 2

Aim: Conversion from different formats to HOURS format.

Step 1:

Make sure you are using the latest version of **Python**.

Step 2:

Install all the following modules before performing the practical:

Commands For the modules:

```
pip install pandas
pip install pillow
pip install scipy==1.2.1
pip install matplotlib
pip install opencv-python
pip install opencv-contrib-python
pip install horus
```

a) Text-Delimited to HORUS

Code:

```
#=====================================================
import pandas as pd
# Input Agreement ============================================
sInputFileName='VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
# Processing Rules ===========================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values ================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ===========================================
OutputData=ProcessData
sOutputFileName='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
0                   Afghanistan         AF         AFG        4
1                 Aland Islands         AX         ALA      248
2                       Albania         AL         ALB        8
3                       Algeria         DZ         DZA       12
4                American Samoa         AS         ASM       16
..                          ...        ...         ...      ...
242  Wallis and Futuna Islands         WF         WLF      876
243              Western Sahara         EH         ESH      732
244                       Yemen         YE         YEM      887
245                      Zambia         ZM         ZMB      894
246                    Zimbabwe         ZW         ZWE      716

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

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

b) XML to HORUS

Code:

```
import pandas as pd
import xml.etree.ElementTree as ET
def df2xml(data):
    header = data.columns
    root = ET.Element('root')
    for row in range(data.shape[0]):
        entry = ET.SubElement(root,'entry')
        for index in range(data.shape[1]):
            schild=str(header[index])
            child = ET.SubElement(entry, schild)
            if str(data[schild][row]) != 'nan':
                child.text = str(data[schild][row])
            else:
                child.text = 'n/a'
            entry.append(child)
    result = ET.tostring(root)
    return result
def xml2df(xml_data):
    root = ET.XML(xml_data)
    all_records = []
```

```
    for i, child in enumerate(root):
        record = {}
        for subchild in child:
            record[subchild.tag] = subchild.text
        all_records.append(record)
    return pd.DataFrame(all_records)
sInputFileName='VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()
print('============================Input Data Values
======================')
print(InputData)
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('==================Process Data Values ====================')
print(ProcessData)
OutputData=ProcessData
sOutputFileName='VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=======================================================')
print('XML to HORUS - Done')
print('=======================================================')
```

Output:

```
============================Input Data Values =============
```

Squeezed text (707 lines).

```
==================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
=======================================================
```

c) JSON to HORUS

Code:

```
import pandas as pd
# Input Agreement ========================================
sInputFileName='VKHCG/05-DS/9999-Data/Country_Code.json'
InputData=pd.read_json(sInputFileName,
                       orient='index',
                       encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
# 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)
# Output Agreement ========================================
OutputData=ProcessData
sOutputFileName='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
10                      Argentina         AR        ARG       32
100                       Hungary         HU        HUN      348
101                       Iceland         IS        ISL      352
..                            ...        ...        ...      ...
95                         Guyana         GY        GUY      328
96                          Haiti         HT        HTI      332
97      Heard and Mcdonald Islands         HM        HMD      334
98   Holy Seeÿ(Vatican City State)         VA        VAT      336
99                         Honduras         HN        HND      340

[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) Database to HORUS

Code:

```
import pandas as pd
import sqlite3 as sq
# Input Agreement ============================================
sInputFileName='VKHCG/05-DS/9999-Data/utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)
print('Input Data Values ===================================')
print(InputData)
# 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)
# Output Agreement ===========================================
OutputData=ProcessData
sOutputFileName='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
>>>
```

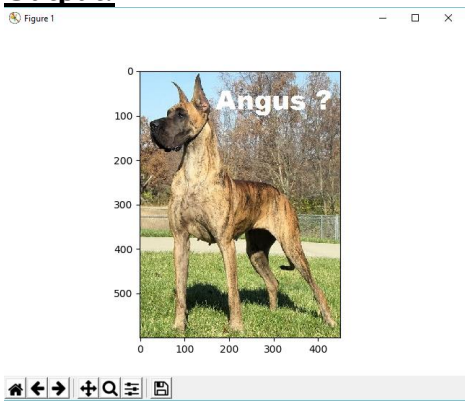
e) Picture to HORUS

Code:

```
from scipy.misc import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Input Agreement ============================================
sInputFileName='VKHCG/05-DS/9999-Data/Angus.jpg'
InputData = imread(sInputFileName, flatten=False, mode='RGBA')
print('Input Data Values ===================================')
print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])
print('=====================================================')
# Processing Rules ===========================================
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
```

```
sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']
ProcessData.columns=sColumns
ProcessData.index.names =['ID']
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
print('=================Process Data Values ===============')
plt.imshow(InputData)
plt.show()
# Output Agreement ============================================
OutputData=ProcessData
print('Storing File')
sOutputFileName='VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Picture to HORUS - Done')
# Utility done ================================================
```

Output:




```
>>>
 RESTART: D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\PICTURE2
HORUS.py

Warning (from warnings module):
  File "D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\PICTURE2HO
RUS.py", line 10
    InputData = imread(sInputFileName, flatten=False, mode='RGBA')
DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
Input Data Values ===================================
X:  600
Y:  450
RGBA:  4
=====================================================
Rows:  180000
Columns : 6
=================Process Data Values ===============
Storing File
=====================================================
Picture to HORUS - Done
>>>
```

f) Video to HORUS

1. Movie to Frames

Code:

```
import os
import shutil
import cv2
sInputFileName='VKHCG/05-DS/9999-Data/dog.mp4'
sDataBaseDir='VKHCG/05-DS/9999-Data/temp'
if os.path.exists(sDataBaseDir):
    shutil.rmtree(sDataBaseDir)
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
print('====================================================')
print('Start Movie to Frames')
print('====================================================')
vidcap = cv2.VideoCapture(sInputFileName)
success,image = vidcap.read()
count = 0
while success:
    success,image = vidcap.read()
    sFrame=sDataBaseDir + str('/dog-frame-' + str(format(count, '04d')) + '.jpg')
    print('Extracted: ', sFrame)
    cv2.imwrite(sFrame, image)
    if os.path.getsize(sFrame) == 0:
        count += -1
        os.remove(sFrame)
        print('Removed: ', sFrame)
    if cv2.waitKey(10) == 27: # exit if Escape is hit
        break
    if count > 100: # exit
        break
    count += 1
print('====================================================')
print('Generated : ', count, ' Frames')
print('====================================================')
print('Movie to Frames HORUS - Done')
print('====================================================')
```

Output:

```
====================================================
Start Movie to Frames
====================================================
Extracted:  VKHCG/05-DS/9999-Data/temp/dog-frame-0000.jpg
Extracted:  VKHCG/05-DS/9999-Data/temp/dog-frame-0001.jpg
Extracted:  VKHCG/05-DS/9999-Data/temp/dog-frame-0002.jpg
Extracted:  VKHCG/05-DS/9999-Data/temp/dog-frame-0003.jpg
Extracted:  VKHCG/05-DS/9999-Data/temp/dog-frame-0004.jpg
Extracted:  VKHCG/05-DS/9999-Data/temp/dog-frame-0100.jpg
Extracted:  VKHCG/05-DS/9999-Data/temp/dog-frame-0101.jpg
====================================================
Generated :  101  Frames
====================================================
Movie to Frames HORUS - Done
====================================================
```

2. **Frames to Horus**

Code:

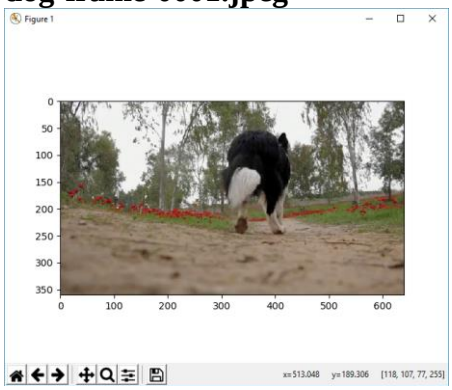
```
from scipy.misc import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
sDataBaseDir='VKHCG/05-DS/9999-Data/temp'
f=0
for file in os.listdir(sDataBaseDir):
    if file.endswith(".jpg"):
        f += 1
        sInputFileName=os.path.join(sDataBaseDir, file)
        print('Process : ', sInputFileName)
        InputData = imread(sInputFileName, flatten=False, mode='RGBA')
        print('Input Data Values ===================================')
        print('X: ',InputData.shape[0])
        print('Y: ',InputData.shape[1])
        print('RGBA: ', InputData.shape[2])
        print('=====================================================')
        ProcessRawData=InputData.flatten()
        y=InputData.shape[2] + 2
        x=int(ProcessRawData.shape[0]/y)
        ProcessFrameData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
        ProcessFrameData['Frame']=file
        print('==============Process Data Values ==================')
        plt.imshow(InputData)
        plt.show()
        if f == 1:
            ProcessData=ProcessFrameData
        else:
            ProcessData=ProcessData.append(ProcessFrameData)
if f > 0:
    sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha','FrameName']
    ProcessData.columns=sColumns
    ProcessFrameData.index.names =['ID']
    print('Rows: ',ProcessData.shape[0])
    print('Columns :',ProcessData.shape[1])
    OutputData=ProcessData
    print('Storing File')
    sOutputFileName='VKHCG/05-DS/9999-Data/HORUS-Movie-Frame.csv'
    OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Processed ; ', f,' frames')
print('=====================================================')
print('Movie to HORUS - Done')
print('=====================================================')
```

Output:

```
=====================================================
==============Process Data Values ==================
Rows:  15667200
Columns : 7
Storing File
=====================================================
Processed ;  102  frames
=====================================================
Movie to HORUS - Done
```

dog-frame-0000.jpeg

dog-frame-0001.jpeg



dog-frame-0000.jpeg

dog-frame-0001.jpeg

g) Audio to HORUS

Code:

```
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='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='VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
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='VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='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='VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='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='VKHCG/05-DS/9999-Data/HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('====================================================')
print('Audio to HORUS - Done')
```

Output:

```
====================================================
Processing :  VKHCG/05-DS/9999-Data/2ch-sound.wav
====================================================
----------------
Audio: 2 channel
----------------
Rate: 22050
----------------
shape: (29016, 2)
dtype: int16
min, max: -16384 14767
----------------
====================================================
Processing :  VKHCG/05-DS/9999-Data/4ch-sound.wav
====================================================
----------------
Audio: 4 channel
----------------
Rate: 44100
----------------
shape: (169031, 4)
dtype: int16
min, max: -31783 26018
----------------
Warning (from warnings module):
  File "C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\lib\tkinter\__init__.py", line 1705
    return self.func(*args)
UserWarning: Creating legend with loc="best" can be slow with large amounts of data.

Warning (from warnings module):
  File "C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\lib\tkinter\__init__.py", line 749
    func(*args)
UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
====================================================
Processing :  VKHCG/05-DS/9999-Data/6ch-sound.wav
====================================================
----------------
Audio: 6 channel
----------------
Rate: 44100
----------------
shape: (257411, 6)
dtype: int16
min, max: -10018 10957
----------------
====================================================
Processing :  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
====================================================
>>>
```

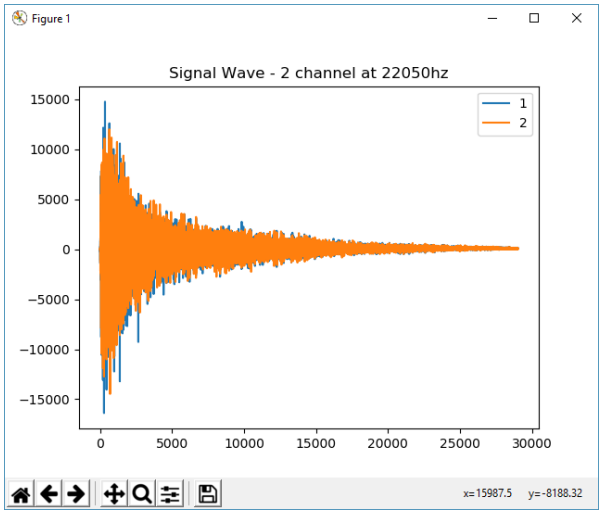


Figure 1
Signal Wave - 2 channel at 22050hz
15000
10000
5000
0
−5000
−10000
−15000
0
5000
10000
15000
20000
25000
30000
1
2
x=15987.5 y=-8188.32

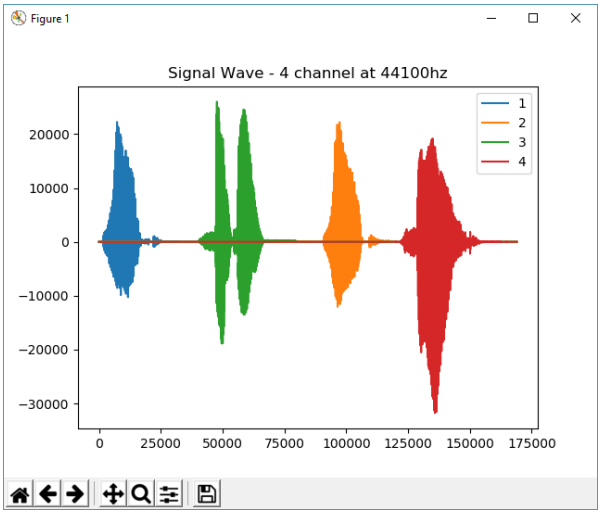


Figure 1
Signal Wave - 4 channel at 44100hz
20000
10000
0
−10000
−20000
−30000
0
25000
50000
75000
100000
125000
150000
175000
1
2
3
4

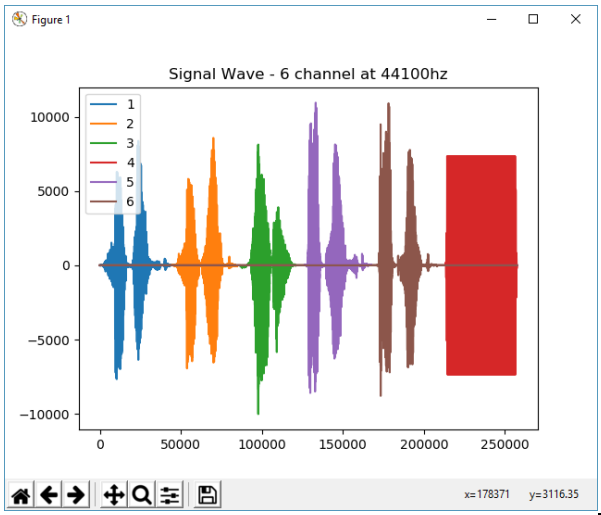


Figure 1
Signal Wave - 6 channel at 44100hz
10000
5000
0
−5000
−10000
0
50000
100000
150000
200000
250000
1
2
3
4
5
6
x=178371 y=3116.35

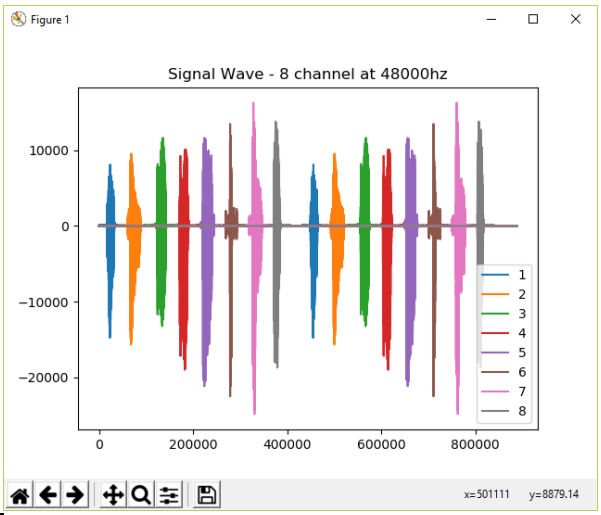


Figure 1
Signal Wave - 8 channel at 48000hz
10000
0
−10000
−20000
0
200000
400000
600000
800000
1
2
3
4
5
6
7
8
x=501111 y=8879.14

Practical No. 3

Aim: Utilities and Auditing

Step 1:
Make sure you are using the latest version of **Python**.

Step 2:
Install all the following modules before performing the practical:

Commands For the modules:

pip install datetime
pip install pandas
pip install matplotlib

A. Fixers Utilities:

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

Code:

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

Output:

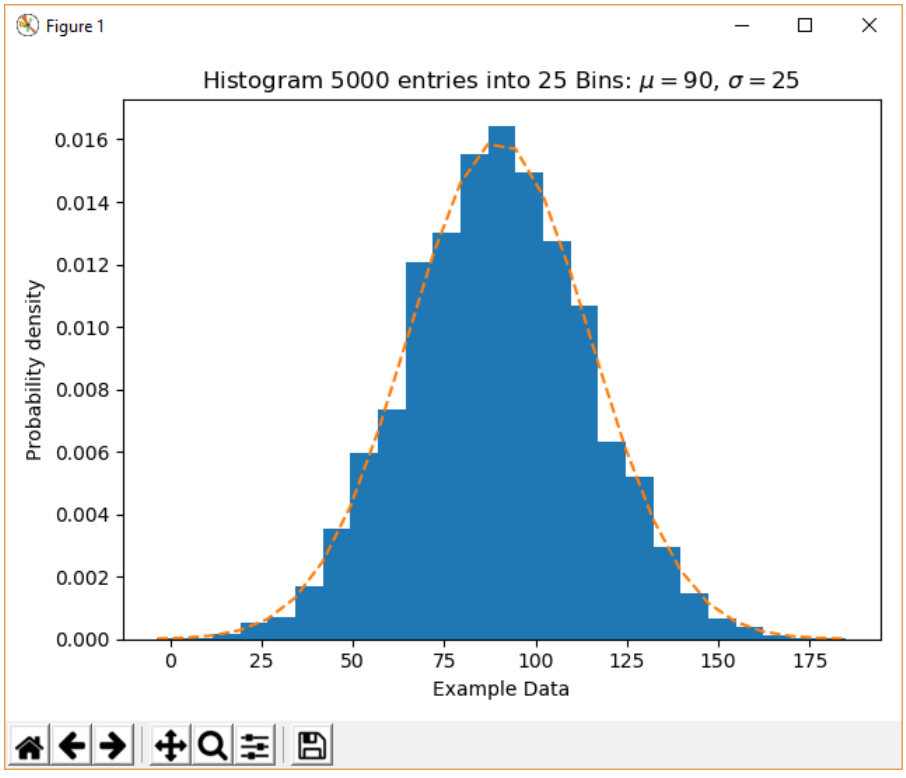
```
#1 Removing leading or lagging spaces from a data entry
>  Data Science with too many spaces is bad!!!  <
> Data Science with too many spaces is bad!!! <
#2 Removing nonprintable characters from a data entry
Bad Data :  Data Science with  funny characters is bad!!!
Clean Data :  D a t a S c i e n c e   w i t h   f u n n y   c h a r a c t e r s
i s   b a d ! ! !
# 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)
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()
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='VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png'
fig.savefig(sPathFig)
plt.show()
```

Output:



Practical No. 4

Aim: Retrieve Superstep

A. Perform the following data processing using R.

Code:

```
library(readr)
IP_DATA_ALL <- read_csv("C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
View(IP_DATA_ALL)
spec(IP_DATA_ALL)
library(tibble)
set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)
IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = TRUE)
View(IP_DATA_ALL_FIX)
sapply(IP_DATA_ALL_FIX, typeof)
library(data.table)
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=hist_country_with_id=rowid_to_column(hist_country, var =
"RowIDCountry")
View(hist_country_fix)
IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_FIX, table(Country)))
View(IP_DATA_COUNTRY_FREQ)
sapply(IP_DATA_ALL_FIX[,'Latitude'], min, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX[,'Country'], min, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX[,'Latitude'], max, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX[,'Country'], max, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [,'Latitude'], mean, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [,'Latitude'], median, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [,'Latitude'], range, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [,'Latitude'], quantile, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [,'Latitude'], sd, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [,'Longitude'], sd, na.rm=TRUE)
```

Output:

```
> library(readr)
> IP_DATA_ALL <- read_csv("C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
Parsed with column specification:
cols(
  X1 = col_double(),
  ID = col_double(),
  Country = col_character(),
  Place.Name = col_character(),
  Post.Code = col_logical(),
  Latitude = col_double(),
  Longitude = col_double(),
  First.IP.Number = col_double(),
  Last.IP.Number = col_double()
)
```

```
|==========================================================| 100%   93 MB
Warning: 1150322 parsing failures.
 row       col           expected actual                                                 file
2312 Post.Code 1/0/T/F/TRUE/FALSE  15222 'C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
2313 Post.Code 1/0/T/F/TRUE/FALSE  15222 'C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
2314 Post.Code 1/0/T/F/TRUE/FALSE  15222 'C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
2315 Post.Code 1/0/T/F/TRUE/FALSE  15222 'C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
2316 Post.Code 1/0/T/F/TRUE/FALSE  15222 'C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
.... ......... ................. ...... .............................................
See problems(...) for more details.

Warning message:
Missing column names filled in: 'X1' [1]
> View(IP_DATA_ALL)
```

Data: IP_DATA_ALL

	X1	ID	Country	Place.Name
1	1	1	BW	Gaborone
2	2	2	BW	Gaborone
3	3	3	BW	Gaborone
4	4	4	BW	Gaborone
5	5	5	BW	Gaborone

```
> spec(IP_DATA_ALL)
cols(
  X1 = col_double(),
  ID = col_double(),
  Country = col_character(),
  Place.Name = col_character(),
  Post.Code = col_logical(),
  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: 1,247,502 x 9
      X1    ID Country Place.Name Post.Code Latitude Longitude First.IP.Number
   <dbl> <dbl> <chr>   <chr>      <lgl>        <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       693078272
 7     7     7 BW      Gaborone   NA           -24.6      25.9       693608448
 8     8     8 BW      Gaborone   NA           -24.6      25.9       696929792
 9     9     9 BW      Gaborone   NA           -24.6      25.9       700438784
10    10    10 BW      Gaborone   NA           -24.6      25.9       702075904
# ... with 1,247,492 more rows, and 1 more variable: Last.IP.Number <dbl>
> IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = TRUE)
> View(IP_DATA_ALL_FIX)
```

Data: IP_DATA_ALL_FIX

	X1	ID	Country	Place.Name
1	1	1	BW	Gaborone
2	2	2	BW	Gaborone
3	3	3	BW	Gaborone
4	4	4	BW	Gaborone
5	5	5	BW	Gaborone

```
> sapply(IP_DATA_ALL_FIX, typeof)
            X1             ID        Country     Place.Name      Post.Code
      "double"       "double"     "character"   "character"       "logical"
      Latitude      Longitude First.IP.Number  Last.IP.Number
      "double"       "double"       "double"       "double"
```

```
> library(data.table)
> 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=hist_country_with_id=rowid_to_column(hist_country, var = "RowIDCountry")
> View(hist_country_fix)
```

Data: hist_country_...

	RowIDCountry	Country
1	1	AD
2	2	AE
3	3	AF
4	4	AG
5	5	AI
6	6	AL
7	7	AM

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

Data: IP_D...

	Country	N
1	AD	46
2	AE	1793
3	AF	15
4	AG	21
5	AI	9
6	AL	91
7	AM	92

```
> sapply(IP_DATA_ALL_FIX[,'Latitude'], min, na.rm=TRUE)
Latitude
-54.2767
> 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
39.58496
> sapply(IP_DATA_ALL_FIX [,'Latitude'], median, na.rm=TRUE)
Latitude
 41.9232
> sapply(IP_DATA_ALL_FIX [,'Latitude'], range, na.rm=TRUE)
     Latitude
[1,] -54.2767
[2,]  78.2167
> sapply(IP_DATA_ALL_FIX [,'Latitude'], quantile, na.rm=TRUE)
      Latitude
0%   -54.27670
25%   35.88960
50%   41.92320
75%   49.28077
100%  78.21670
> sapply(IP_DATA_ALL_FIX [,'Latitude'], sd, na.rm=TRUE)
Latitude
16.85403
> sapply(IP_DATA_ALL_FIX [,'Longitude'], sd, na.rm=TRUE)
Longitude
 71.78892
>
```

B. Program to retrieve different attributes of data.

Code:

```
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="latin-1")
```

Output:

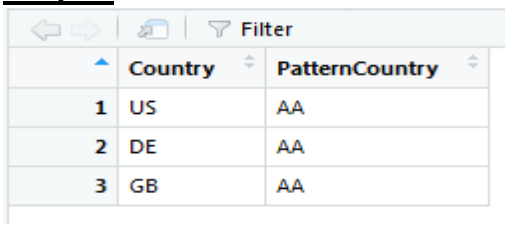
```
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 1247502
Columns: 9
### Raw Data Set #####################################
Unnamed: 0 <class 'str'>
### Fixed Data Set #####################################
ID <class 'str'>
### Fixed Data Set #####################################
Country <class 'str'>
### Fixed Data Set #####################################
Place.Name <class 'str'>
### Fixed Data Set #####################################
Post.Code <class 'str'>
### Fixed Data Set #####################################
Latitude <class 'str'>
### Fixed Data Set #####################################
Longitude <class 'str'>
### Fixed Data Set #####################################
First.IP.Number <class 'str'>
### Fixed Data Set #####################################
Last.IP.Number <class 'str'>
### Fixed Data Set #####################################
Last.IP.Number <class 'str'>
Fixed Data Set with ID
```

C. Data Pattern

Code:

```
library(readr)
library(data.table)
FileName=paste0('c:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv')
IP_DATA_ALL <- read_csv(FileName)
hist_country=data.table(Country=unique(IP_DATA_ALL$Country))
pattern_country=data.table(Country=hist_country$Country,
PatternCountry=hist_country$Country)
oldchar=c(letters,LETTERS)
newchar=replicate(length(oldchar),"A")
for (r in seq(nrow(pattern_country))){
 s=pattern_country[r,]$PatternCountry;
 for (c in seq(length(oldchar))){
  s=chartr(oldchar[c],newchar[c],s)
};
for (n in seq(0,9,1)){
 s=chartr(as.character(n),"N",s)
};
 s=chartr(" ","b",s)
 s=chartr(".","u",s)
 pattern_country[r,]$PatternCountry=s;
};
View(pattern_country)
```

Output:



Filter

	Country	PatternCountry
1	US	AA
2	DE	AA
3	GB	AA

D. Loading IP_DATA_ALL:

Code:

```
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="latin-1")
print('### Done!! ################################################')
```

Output:

```
========================== RESTART: C:/neda/4d.py ==========================
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Unnamed: 0 <class 'str'>
### Fixed Data Set ####################################
ID <class 'str'>
### Fixed Data Set ####################################
Country <class 'str'>
### Fixed Data Set ####################################
Place.Name <class 'str'>
### Fixed Data Set ####################################
Post.Code <class 'str'>
### Fixed Data Set ####################################
Latitude <class 'str'>
### Fixed Data Set ####################################
Longitude <class 'str'>
### Fixed Data Set ####################################
First.IP.Number <class 'str'>
### Fixed Data Set ####################################
Last.IP.Number <class 'str'>
### Fixed Data Set ####################################
Last.IP.Number <class 'str'>
Fixed Data Set with ID
```

Vermeulen PLC
1.Designing a routing diagram for company

Code:

```
import sys
import os
import pandas as pd
from math import radians, cos, sin, asin, sqrt
def haversine(lon1, lat1, lon2, lat2,stype):
"""
Calculate the great circle distance between two points
on the earth (specified in decimal degrees)
"""
# convert decimal degrees to radians
lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
# haversine formula
dlon = lon2 - lon1
dlat = lat2 - lat1
a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
c = 2 * asin(sqrt(a))
if stype == 'km':
  r = 6371 # Radius of earth in kilometers
else:
  r = 3956 # Radius of earth in miles
d=round(c * r,3)
return d
Base='C:/VKHCG'
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_CORE.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
IP_DATA = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
IP_DATA.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA1 = IP_DATA
IP_DATA1.insert(0, 'K', 1)
IP_DATA2 = IP_DATA1
print(IP_DATA1.shape)
IP_CROSS=pd.merge(right=IP_DATA1,left=IP_DATA2,on='K')
IP_CROSS.drop('K', axis=1, inplace=True)
IP_CROSS.rename(columns={'Longitude_x': 'Longitude_from', 'Longitude_y':
'Longitude_to'}, inplace=True)
IP_CROSS.rename(columns={'Latitude_x': 'Latitude_from', 'Latitude_y': 'Latitude_to'},
inplace=True)
```

```
IP_CROSS.rename(columns={'Place_Name_x': 'Place_Name_from', 'Place_Name_y':
'Place_Name_to'}, inplace=True)
IP_CROSS.rename(columns={'Country_x': 'Country_from', 'Country_y': 'Country_to'},
inplace=True)
IP_CROSS['DistanceBetweenKilometers'] = IP_CROSS.apply(lambda row:
 haversine(
        row['Longitude_from'],
        row['Latitude_from'],
        row['Longitude_to'],
        row['Latitude_to'],
        'km')
        ,axis=1)
IP_CROSS['DistanceBetweenMiles'] = IP_CROSS.apply(lambda row:
 haversine(
        row['Longitude_from'],
        row['Latitude_from'],
        row['Longitude_to'],
        row['Latitude_to'],
        'miles')
        ,axis=1)
print(IP_CROSS.shape)
sFileName2=sFileDir + '/Retrieve_IP_Routing.csv'
IP_CROSS.to_csv(sFileName2, index = False, encoding="latin-1")
print('### Done!! ################################################')
```

Output:

1	Country_from	Place_Name_from	Latitude_from	Longitude_from	Country_to	Place_Name_to	Latitude_to	Longitude_to	DistanceBetweenKilometers	DistanceBetweenMiles
2	US	New York	40.7528	-73.9725	US	New York	40.7528	-73.9725	0	0
3	US	New York	40.7528	-73.9725	US	New York	40.7214	-74.0052	4.448	2.762
4	US	New York	40.7528	-73.9725	US	New York	40.7662	-73.9862	1.885	1.17
5	US	New York	40.7528	-73.9725	US	New York	40.7449	-73.9782	1.001	0.622
6	US	New York	40.7528	-73.9725	US	New York	40.7605	-73.9933	1.95	1.211
7	US	New York	40.7528	-73.9725	US	New York	40.7588	-73.968	0.767	0.476
8	US	New York	40.7528	-73.9725	US	New York	40.7637	-73.9727	1.212	0.753
9	US	New York	40.7528	-73.9725	US	New York	40.7553	-73.9924	1.699	1.055
10	US	New York	40.7528	-73.9725	US	New York	40.7308	-73.9975	3.228	2.004
11	US	New York	40.7528	-73.9725	US	New York	40.7694	-73.9609	2.088	1.297

2.Building a Diagram for the Scheduling of Jobs

Code:

```
import sys
import os
import pandas as pd
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
   Base='C:/VKHCG'
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','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
##########################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])
sFileName2=sFileDir + '/' + OutputFileName
ROUTERLOC.to_csv(sFileName2, index = False, encoding="latin-1")
print('### Done!! ############################################')
```

Output:

```
=== RESTART: C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-Router-Location.py ===
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
Rows : 150
Columns : 4
### Done!! ############################################
>>>
```

A1 fx Country

	A	B	C	D	E
1	Country	Place_Nar	Latitude	Longitude	
2	US	New York	40.7528	-73.9725	
3	US	New York	40.7214	-74.0052	
4	US	New York	40.7662	-73.9862	
5	US	New York	40.7449	-73.9782	
6	US	New York	40.7605	-73.9933	
7	US	New York	40.7588	-73.968	
8	US	New York	40.7637	-73.9727	
9	US	New York	40.7553	-73.9924	
10	US	New York	40.7308	-73.9975	
11	US	New York	40.7694	-73.9609	
12	US	New York	40.733	-74.0078	
13	US	New York	40.7505	-73.9931	

Krennwallner AG
1.Picking Content for Billboards

Code:

```
import sys
import os
import pandas as pd
InputFileName='DE_Billboard_Locations.csv'
OutputFileName='Retrieve_DE_Billboard_Locations.csv'
Company='02-Krennwallner'
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
Base='C:/VKHCG'
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','PlaceName','Latitude','Longitude'])
IP_DATA_ALL.rename(columns={'PlaceName': 'Place_Name'}, inplace=True)
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])
sFileName2=sFileDir + '/' + OutputFileName
ROUTERLOC.to_csv(sFileName2, index = False)
print('### Done!! ############################################')
```

Output:

```
========================== RESTART: C:/neda/4d2.py ==========================
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/02-Krennwallner/00-RawData/DE_Billboard_Locations.csv
Rows : 8873
Columns : 4
### Done!! ############################################
>>>
```

	A	B	C	D	E
1	Country	Place_Nar	Latitude	Longitude	
2	DE	Lake	51.7833	8.5667	
3	DE	Horb	48.4333	8.6833	
4	DE	Hardenbe	51.1	7.7333	
5	DE	Horn-bad	51.9833	8.9667	
6	DE	Winkel	51.55	13.3833	
7	DE	Rohrdorf	47.7333	10.0833	
8	DE	Sch<f6>ne	51.9833	12.8333	

2.Understand your online Visitor Data

Code:

```
import sys
import os
import pandas as pd
import gzip as gz
InputFileName='IP_DATA_ALL.csv'
OutputFileName='Retrieve_Online_Visitor'
CompanyIn= '01-Vermeulen'
CompanyOut= '02-Krennwallner'
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
Base='C:/VKHCG'
sFileName=Base + '/' + CompanyIn + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','Place Name','Latitude','Longitude','First IP Number','Last IP Number'])
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA_ALL.rename(columns={'First IP Number': 'First_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Last IP Number': 'Last_IP_Number'}, inplace=True)
################################################################
sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
visitordata = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
visitordata10=visitordata.head(10)
print('Rows :',visitordata.shape[0])
print('Columns :',visitordata.shape[1])
print('Export CSV')
sFileName2=sFileDir + '/' + OutputFileName + '.csv'
visitordata.to_csv(sFileName2, index = False)
print('Store All:',sFileName2)
sFileName3=sFileDir + '/' + OutputFileName + '_10.csv'
visitordata10.to_csv(sFileName3, index = False)
print('Store 10:',sFileName3)
for z in ['gzip', 'bz2', 'xz']:
    if z == 'gzip':
        sFileName4=sFileName2 + '.gz'
    else:
        sFileName4=sFileName2 + '.' + z
    visitordata.to_csv(sFileName4, index = False, compression=z)
    print('Store :',sFileName4)
print('Export JSON')
```

```
for sOrient in ['split','records','index', 'columns','values','table']:
    sFileName2=sFileDir + '/' + OutputFileName + '_' + sOrient + '.json'
    visitordata.to_json(sFileName2,orient=sOrient,force_ascii=True)
    print('Store All:',sFileName2)
    sFileName3=sFileDir + '/' + OutputFileName + '_10_' + sOrient + '.json'
    visitordata10.to_json(sFileName3,orient=sOrient,force_ascii=True)
    print('Store 10:',sFileName3)
    sFileName4=sFileName2 + '.gz'
    file_in = open(sFileName2, 'rb')
    file_out = gz.open(sFileName4, 'wb')
    file_out.writelines(file_in)
    file_in.close()
    file_out.close()
    print('Store GZIP All:',sFileName4)
    sFileName5=sFileDir + '/' + OutputFileName + '_' + sOrient + '_UnGZip.json'
    file_in = gz.open(sFileName4, 'rb')
    file_out = open(sFileName5, 'wb')
    file_out.writelines(file_in)
    file_in.close()
    file_out.close()
    print('Store UnGZIP All:',sFileName5)
print('### Done!! ################################################')
```

Output:

```
== RESTART: C:\VKHCG\02-Krennwallner\01-Retrieve\Retrieve-Online-Visitor.py ==
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows : 3562
Columns : 6
Export CSV
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10.csv
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.gz
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.bz2
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.xz
Export JSON
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_split.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_records.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_index.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_columns.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_values.json
```

A1 fx Country

	A	B	C	D	E	F
1	Country	Place_Nar	Latitude	Longitude	First_IP_N	Last_IP_Nu
2	BW	Gaborone	-24.6464	25.9119	6.93E+08	6.93E+08
3	BW	Gaborone	-24.6464	25.9119	6.93E+08	6.93E+08
4	BW	Gaborone	-24.6464	25.9119	6.93E+08	6.93E+08
5	BW	Gaborone	-24.6464	25.9119	6.93E+08	6.93E+08
6	BW	Gaborone	-24.6464	25.9119	6.93E+08	6.93E+08
7	BW	Gaborone	-24.6464	25.9119	6.93E+08	6.93E+08
8	BW	Gaborone	-24.6464	25.9119	6.94E+08	6.94E+08
9	BW	Gaborone	-24.6464	25.9119	6.97E+08	6.97E+08
10	BW	Gaborone	-24.6464	25.9119	7E+08	7E+08

Hillman Ltd
1.Planning Shipping Rules for best fit international logistics
EXW

Code:

```
import os
import sys
import pandas as pd
IncoTerm='EXW'
InputFileName='Incoterm_2010.csv'
OutputFileName='Retrieve_Incoterm_' + IncoTerm + '_RuleSet.csv'
Company='03-Hillman'
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
  os.makedirs(sFileDir)
### Import Incoterms
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('############')
print('Loading :',sFileName)
IncotermGrid=pd.read_csv(sFileName,header=0,low_memory=False)
IncotermRule=IncotermGrid[IncotermGrid.Shipping_Term == IncoTerm]
print('Rows :',IncotermRule.shape[0])
print('Columns :',IncotermRule.shape[1])
print('############')
print(IncotermRule)
sFileName=sFileDir + '/' + OutputFileName
IncotermRule.to_csv(sFileName, index = False)
print('### Done!! ############################################')
```

Output:

```
===== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Incoterm-EXW.py =====
################################
Working Base : C:/VKHCG  using  win32
################################
############
Loading : C:/VKHCG/03-Hillman/00-RawData/Incoterm_2010.csv
Rows : 1
Columns : 9
############
  Shipping_Term  Seller Carrier Port_From  ... Port_To Terminal Named_Place  Buy
er
0           EXW  Seller   Buyer     Buyer  ...   Buyer    Buyer       Buyer  Buy
er

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

2.Adopt new shipping containers

Code:

```
import sys
import os
import pandas as pd
ContainerFileName='Retrieve_Container.csv'
BoxFileName='Retrieve_Box.csv'
ProductFileName='Retrieve_Product.csv'
Company='03-Hillman'
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
### Create the Containers
containerLength=range(1,21)
containerWidth=range(1,10)

containerHeigth=range(1,6)
containerStep=1
c=0
for l in containerLength:
    for w in containerWidth:
        for h in containerHeigth:
            containerVolume=(l/containerStep)*(w/containerStep)*(h/containerStep)
            c=c+1
ContainerLine=[('ShipType', ['Container']),
               ('UnitNumber', ('C'+format(c,"06d"))),
               ('Length',(format(round(l,3),".4f"))),
              ('Width',(format(round(w,3),".4f"))),
              ('Height',(format(round(h,3),".4f"))),
              ('ContainerVolume',(format(round(containerVolume,6),".6f")))]
if c==1:
       ContainerFrame = pd.DataFrame.from_dict(ContainerLine)
else:
      ContainerRow = pd.DataFrame.from_dict(ContainerLine)
     ContainerFrame = ContainerFrame.append(ContainerRow)
  ContainerFrame.index.name = 'IDNumber'
print('#################')
print('## Container')
print('#################')
print('Rows :',ContainerFrame.shape[0])
print('Columns :',ContainerFrame.shape[1])
```

```
print('#################')
sFileContainerName=sFileDir + '/' + ContainerFileName
ContainerFrame.to_csv(sFileContainerName, index = False)
## Create valid Boxes with packing foam
boxLength=range(1,21)
boxWidth=range(1,21)
boxHeigth=range(1,21)
packThick=range(0,6)
boxStep=10
b=0
for l in boxLength:
    for w in boxWidth:
        for h in boxHeigth:
            for t in packThick:
                boxVolume=round((l/boxStep)*(w/boxStep)*(h/boxStep),6)
                productVolume=round(((l-t)/boxStep)*((w-t)/boxStep)*((h-t)/boxStep),6)
                if productVolume > 0:
                b=b+1
                 BoxLine=[('ShipType', ['Box']),
             ('UnitNumber', ('B'+format(b,"06d"))),
            ('Length',(format(round(l/10,6),".6f"))),
            ('Width',(format(round(w/10,6),".6f"))),
            ('Height',(format(round(h/10,6),".6f"))),
             ('Thickness',(format(round(t/5,6),".6f"))),
              ('BoxVolume',(format(round(boxVolume,9),".9f"))),
               ('ProductVolume',(format(round(productVolume,9),".9f")))]
  if b==1:
                BoxFrame = pd.DataFrame.from_dict(BoxLine)
  else:
                BoxRow = pd.DataFrame.from_dict(BoxLine)
BoxFrame = BoxFrame.append(BoxRow)
BoxFrame.index.name = 'IDNumber'
print('#################')
print('## Box')
print('#################')
print('Rows :',BoxFrame.shape[0])
print('Columns :',BoxFrame.shape[1])
print('#################')
sFileBoxName=sFileDir + '/' + BoxFileName
BoxFrame.to_csv(sFileBoxName, index = False)
## Create valid Product
productLength=range(1,21)
productWidth=range(1,21)
productHeigth=range(1,21)
productStep=10
p=0
```

```
for l in productLength:
for w in productWidth:
for h in productHeigth:
productVolume=round((l/productStep)*(w/productStep)*(h/productStep),6)
if productVolume > 0:
  p=p+1
  ProductLine=[('ShipType', ['Product']),
   ('UnitNumber', ('P'+format(p,"06d"))),
   ('Length',(format(round(l/10,6),".6f"))),
    ('Width',(format(round(w/10,6),".6f"))),
     ('Height',(format(round(h/10,6),".6f"))),
    ('ProductVolume',(format(round(productVolume,9),".9f")))]
if p==1:
    ProductFrame = pd.DataFrame.from_dict(ProductLine)
else:
   ProductRow = pd.DataFrame.from_dict(ProductLine)
   ProductFrame = ProductFrame.append(ProductRow)
BoxFrame.index.name = 'IDNumber'
print('##################')
print('## Product')
print('##################')
print('Rows :',ProductFrame.shape[0])
print('Columns :',ProductFrame.shape[1])
print('##################')
sFileProductName=sFileDir + '/' + ProductFileName
ProductFrame.to_csv(sFileProductName, index = False)
print('### Done!! ################################################')
```

Output

```
==== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Container-Plan.py ====
################################
Working Base : C:/VKHCG  using  win32
################################
#################
## Container
#################
Rows : 5400
Columns : 2
#################
#################
## Box
#################
Rows : 275880
Columns : 2
#################
#################
## Product
#################
Rows : 48000
Columns : 2
#################
### Done!! ############################################
>>>
```

4.Create a Delivery route

Code:

```
import os
import sys
import pandas as pd
from geopy.distance import vincenty
InputFileName='GB_Postcode_Warehouse.csv'
OutputFileName='Retrieve_GB_Warehouse.csv'
Company='03-Hillman'
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('############')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False)
WarehouseClean=Warehouse[Warehouse.latitude != 0]
WarehouseGood=WarehouseClean[WarehouseClean.longitude != 0]
WarehouseGood.drop_duplicates(subset='postcode', keep='first', inplace=True)
WarehouseGood.sort_values(by='postcode', ascending=1)
sFileName=sFileDir + '/' + OutputFileName
WarehouseGood.to_csv(sFileName, index = False)
WarehouseLoop = WarehouseGood.head(20)
for i in range(0,WarehouseLoop.shape[0]):
     print('Run :',i,' ========>>>>>>>>>>>',WarehouseLoop['postcode'][i])
     WarehouseHold = WarehouseGood.head(10000)
    WarehouseHold['Transaction']=WarehouseHold.apply(lambda row:
    'WH-to-WH'
    ,axis=1)
OutputLoopName='Retrieve_Route_' + 'WH-' + WarehouseLoop['postcode'][i] + '_Route.csv'
    WarehouseHold['Seller']=WarehouseHold.apply(lambda row:
   'WH-' + WarehouseLoop['postcode'][i]
    ,axis=1)
```

Output

```
################################
Working Base : C:/VKHCG using win32
################################
############
Loading : C:/VKHCG/03-Hillman/00-RawData/GB_Postcode_Warehouse.csv
Run : 0 ========>>>>>>>>>> AB10
Run : 1 ========>>>>>>>>>> AB11
```

Run : 2 =======>>>>>>>>>> AB12
Run : 3 =======>>>>>>>>>> AB13
Run : 4 =======>>>>>>>>>> AB14
Run : 5 =======>>>>>>>>>> AB15
Run : 6 =======>>>>>>>>>> AB16
Run : 7 =======>>>>>>>>>> AB21
Run : 8 =======>>>>>>>>>> AB22
Run : 9 =======>>>>>>>>>> AB23
Run : 10 =======>>>>>>>>>> AB24
Run : 11 =======>>>>>>>>>> AB25
Run : 12 =======>>>>>>>>>> AB30
Run : 13 =======>>>>>>>>>> AB31
Run : 14 =======>>>>>>>>>> AB32
Run : 15 =======>>>>>>>>>> AB33
Run : 16 =======>>>>>>>>>> AB34
Run : 17 =======>>>>>>>>>> AB35
Run : 18 =======>>>>>>>>>> AB36
Run : 19 =======>>>>>>>>>> AB37
Done!!
>>

	A	B	C	D	E	F	G	H
1	Transactio	Seller	Seller_Lat	Seller_Lor	Buyer	Buyer_Lat	Buyer_Lor	Distance
2	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB10	57.13514	-2.11731	1.024915
3	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB11	57.13875	-2.09089	0
4	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB12	57.101	-2.1106	2.715503
5	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB13	57.10801	-2.23776	5.922893
6	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB14	57.10076	-2.27073	7.26182
7	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB15	57.13868	-2.16525	2.797537
8	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB16	57.16115	-2.15543	2.880038
9	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB21	57.2096	-2.20033	6.399669
10	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB22	57.18724	-2.11913	3.519367
11	WH-to-WI	WH-AB11	57.13875	-2.09089	WH-AB23	57.21242	-2.08776	5.099157

Clark Ltd

Code:

```
import sys
import os
import shutil
import zipfile
import pandas as pd
Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
Company='04-Clark'
ZIPFiles=['Data_female-names','Data_male-names','Data_last-names']
for ZIPFile in ZIPFiles:
    InputZIPFile=Base+'/'+Company+'/00-RawData/' + ZIPFile + '.zip'
    OutputDir=Base+'/'+Company+'/01-Retrieve/01-EDS/02-Python/' + ZIPFile
```

```
    OutputFile=Base+'/'+Company+'/01-Retrieve/01-EDS/02-Python/Retrieve-
'+ZIPFile+'.csv'
    zip_file = zipfile.ZipFile(InputZIPFile, 'r')
    zip_file.extractall(OutputDir)
   zip_file.close()
    t=0
    for dirname, dirnames, filenames in os.walk(OutputDir):
      for filename in filenames:
          sCSVFile = dirname + '/' + filename
          t=t+1
          if t==1:
          NameRawData=pd.read_csv(sCSVFile,header=None,low_memory=False)
          NameData=NameRawData
    else:
         NameRawData=pd.read_csv(sCSVFile,header=None,low_memory=False)
         NameData=NameData.append(NameRawData)
NameData.rename(columns={0 : 'NameValues'},inplace=True)
NameData.to_csv(OutputFile, index = False)
shutil.rmtree(OutputDir)
print('Process: ',InputZIPFile)
print('### Done!! ################################################')
```

Output

```
======= RESTART: C:\VKHCG\04-Clark\01-Retrieve\Retrieve-PersonData.py =======
################################
Working Base : C:/VKHCG  using  win32
################################
Process:  C:/VKHCG/04-Clark/00-RawData/Data_female-names.zip
Process:  C:/VKHCG/04-Clark/00-RawData/Data_male-names.zip
Process:  C:/VKHCG/04-Clark/00-RawData/Data_last-names.zip
### Done!! ############################################
>>>
```

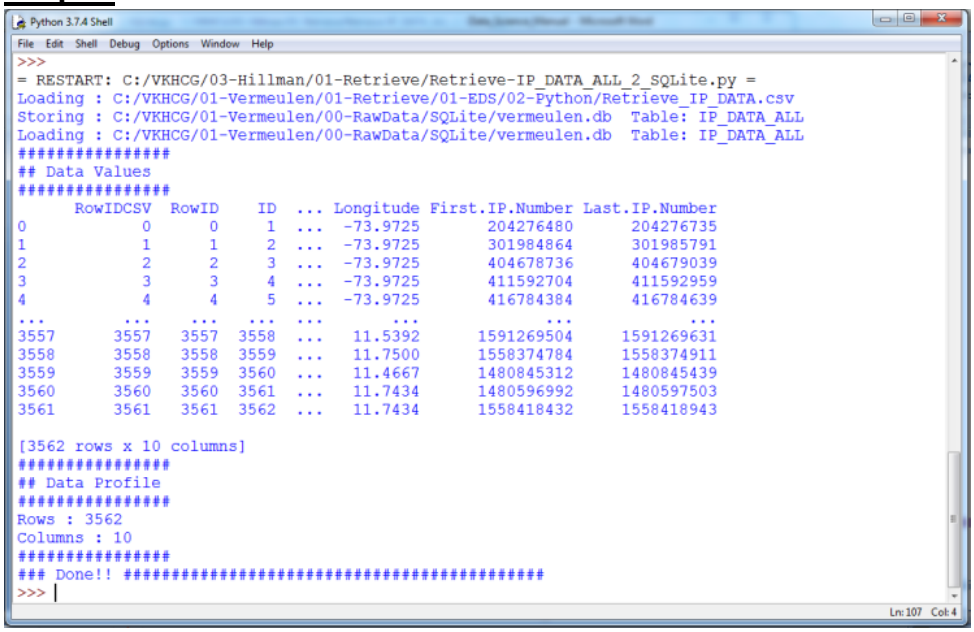
Connecting to Other Data Sources
SQLite:

Code:

```
import sqlite3 as sq
import pandas as pd
Base='C:/VKHCG'
sDatabaseName=Base + '/01-Vermeulen/00-RawData/SQLite/vermeulen.db'
conn = sq.connect(sDatabaseName)
sFileName='C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-
Python/Retrieve_IP_DATA.csv'
print('Loading :',sFileName)
IP_DATA_ALL_FIX=pd.read_csv(sFileName,header=0,low_memory=False)
IP_DATA_ALL_FIX.index.names = ['RowIDCSV']
sTable='IP_DATA_ALL'
print('Storing :',sDatabaseName,' Table:',sTable)
```

```
IP_DATA_ALL_FIX.to_sql(sTable, conn, if_exists="replace")
print('Loading :',sDatabaseName,' Table:',sTable)
TestData=pd.read_sql_query("select * from IP_DATA_ALL;", conn)
print('#################')
print('## Data Values')
print('#################')
print(TestData)
print('#################')
print('## Data Profile')
print('#################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('### Done!! ############################################')
```

Output:



MySQL:

Open MySql , Create a database “DataScience”, Create a python file and add the following code:

Code:

```
################# Connection With MySQL ######################
import mysql.connector
conn = mysql.connector.connect(host='localhost',
database='DataScience',
user='root',
password='root')
conn.connect
if(conn.is_connected):
        print('###### Connection With MySql Established Successfullly ##### ')
else:
```

print('Not Connected -- Check Connection Properites')

Output:

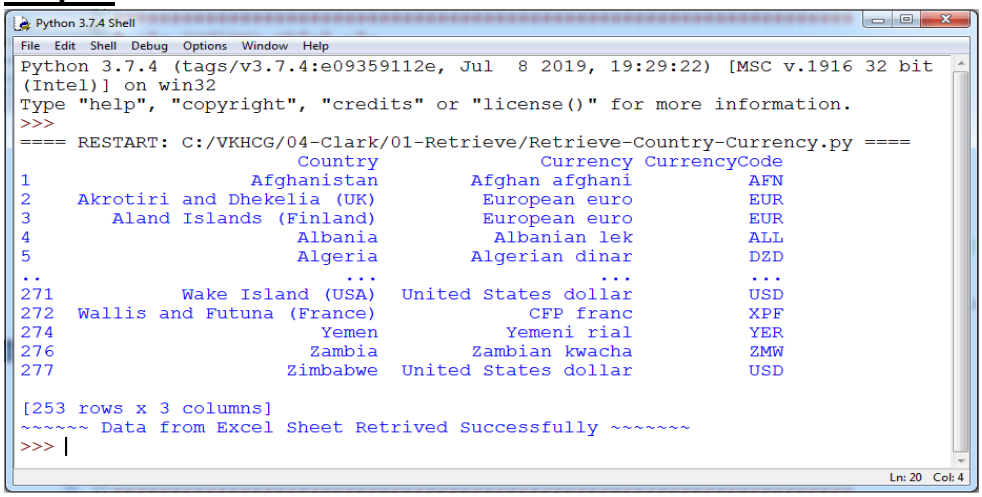
```
>>>
 RESTART: C:/Users/User/AppData/Local/Programs/Python/Python37-32/mysqlconnection.py
###### Connection With MySql Established Successfullly #####
>>>
```

Code:

Microsoft Excel

```
import os
import pandas as pd
Base='C:/VKHCG'
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
#if not os.path.exists(sFileDir):
#os.makedirs(sFileDir)
CurrencyRawData = pd.read_excel('C:/VKHCG/01-Vermeulen/00-
RawData/Country_Currency.xlsx')
sColumns = ['Country or territory', 'Currency', 'ISO-4217']
CurrencyData = CurrencyRawData[sColumns]
CurrencyData.rename(columns={'Country or territory': 'Country', 'ISO-4217':
'CurrencyCode'}, inplace=True)
CurrencyData.dropna(subset=['Currency'],inplace=True)
CurrencyData['Country'] = CurrencyData['Country'].map(lambda x: x.strip())
CurrencyData['Currency'] = CurrencyData['Currency'].map(lambda x:
x.strip())
CurrencyData['CurrencyCode'] = CurrencyData['CurrencyCode'].map(lambda x:
x.strip())
print(CurrencyData)
print('~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~ ')
sFileName=sFileDir + '/Retrieve-Country-Currency.csv'
CurrencyData.to_csv(sFileName, index = False)
```

Output:



Practical No. 5

Aim: Assessing Data

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

i. Drop the Columns Where All Elements Are Missing Values

Code:

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
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('##################################')
#####################################################################
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:

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

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

Code:

```
#####################################################################
# -*- 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:

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

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

Code:

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
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('##################################')
#####################################################################
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:

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

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

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

Code:

```
#####################################################################
import sys
import os
import pandas as pd
#####################################################################
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('#################################')
#####################################################################
sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sInputFileName3='01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv'
#####################################################################
sOutputFileName='Assess-Network-Routing-Company.csv'
Company='01-Vermeulen'
#####################################################################
#####################################################################
### Import Country Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('#################################')
print('Loading :',sFileName)
print('#################################')
CountryData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
print('#################################')
#####################################################################
## Assess Country Data
#####################################################################
print('#################################')
print('Changed :',CountryData.columns.values)
CountryData.rename(columns={'Country': 'Country_Name'}, inplace=True)
CountryData.rename(columns={'ISO-2-CODE': 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inplace=True)
CountryData.drop('ISO-3-Code', axis=1, inplace=True)
CountryData.drop('RowID', axis=1, inplace=True)
print('To :',CountryData.columns.values)
print('#################################')
#####################################################################
#####################################################################
```

```
### Import Company Data
######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('################################')
######################################################################
## Assess Company Data
######################################################################
print('################################')
print('Changed :',CompanyData.columns.values)
CompanyData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CompanyData.columns.values)
print('################################')
######################################################################
######################################################################
### Import Customer Data
######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName3
print('################################')
print('Loading :',sFileName)
print('################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
print('################################')
print('Loaded Customer :',CustomerRawData.columns.values)
print('################################')
######################################################################
CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('################################')
print('Remove Blank Country Code')
print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])
print('################################')
######################################################################
print('################################')
print('Changed :',CustomerData.columns.values)
CustomerData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CustomerData.columns.values)
print('################################')
######################################################################
print('################################')
print('Merge Company and Country Data')
print('################################')
CompanyNetworkData=pd.merge(
    CompanyData,
    CountryData,
    how='inner',
```

```
        on='Country_Code'
        )
#####################################################################
print('###################################')
print('Change ',CompanyNetworkData.columns.values)
for i in CompanyNetworkData.columns.values:
    j='Company_'+i
    CompanyNetworkData.rename(columns={i: j}, inplace=True)
print('To ', CompanyNetworkData.columns.values)
print('###################################')
#####################################################################
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName
print('###################################')
print('Storing :', sFileName)
print('###################################')
CompanyNetworkData.to_csv(sFileName, index = False, encoding="latin-1")
#####################################################################
#####################################################################
print('###################################')
print('### Done!! ######################')
print('###################################')
#####################################################################
```

Output:

Go to C:\VKHCG\01-Vermeulen\02-Assess\01-EDS\02-Python folder and open Assess-Network-Routing-Company.csv

	A	B	C	D	E
1	hy_Countr	Company_Place_Name	Company_Latitude	Company_Longitude	Company_Country_Name
2	US	New York	40.7528	-73.9725	United States of America
3	US	New York	40.7214	-74.0052	United States of America
4	US	New York	40.7662	-73.9862	United States of America
5	US	New York	40.7449	-73.9782	United States of America
6	US	New York	40.7605	-73.9933	United States of America
7	US	New York	40.7588	-73.968	United States of America
8	US	New York	40.7637	-73.9727	United States of America
9	US	New York	40.7553	-73.9924	United States of America

Next, Access the customers location using network router location.

Code:

```
#####################################################################
import sys
import os
import pandas as pd
#####################################################################
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=Base+'/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-
Routing-Customer.csv'
################################################################
sOutputFileName='Assess-Network-Routing-Customer.gml'
Company='01-Vermeulen'
################################################################
################################################################
### Import Country Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
CustomerData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CustomerData.columns.values)
print('##################################')
################################################################
print(CustomerData.head())

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

Output:

Assess-Network-Routing-Customer.csv

	A	B	C	D	E
1	ıer_Countr	tomer_Place_Na	Customer_Latitude	Customer_Longitude	Customer_Country_Name
2	BW	Gaborone	-24.6464	25.9119	Botswana
3	BW	Francistown	-21.1667	27.5167	Botswana
4	BW	Maun	-19.9833	23.4167	Botswana
5	BW	Molepolole	-24.4167	25.5333	Botswana
6	NE	Niamey	13.5167	2.1167	Niger
7	MZ	Maputo	-25.9653	32.5892	Mozambique
8	MZ	Tete	-16.1564	33.5867	Mozambique
9	MZ	Quelimane	-17.8786	36.8883	Mozambique
10	MZ	Chimoio	-19.1164	33.4833	Mozambique
11	MZ	Matola	-25.9622	32.4589	Mozambique
12	MZ	Pemba	-12.9608	40.5078	Mozambique
13	MZ	Lichinga	-13.3128	35.2406	Mozambique
14	MZ	Maxixe	-23.8597	35.3472	Mozambique
15	MZ	Chibuto	-24.6867	33.5306	Mozambique
16	MZ	Ressano Garcia	-25.4428	31.9953	Mozambique
17	GH	Tema	5.6167	-0.0167	Ghana
18	GH	Kumasi	6.6833	-1.6167	Ghana
19	GH	Takoradi	4.8833	-1.75	Ghana
20	GH	Accra	5.55	-0.2167	Ghana

Assess-Network-Routing-Node.py

Code:

```
#####################################################################
import sys
import os
import pandas as pd
#####################################################################
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='01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
#####################################################################
sOutputFileName='Assess-Network-Routing-Node.csv'
Company='01-Vermeulen'
#####################################################################
### Import IP Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
IPData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded IP :', IPData.columns.values)
print('#################################')
#####################################################################
print('#################################')
print('Changed :',IPData.columns.values)
IPData.drop('RowID', axis=1, inplace=True)
IPData.drop('ID', axis=1, inplace=True)
IPData.rename(columns={'Country': 'Country_Code'}, inplace=True)
IPData.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
IPData.rename(columns={'Post.Code': 'Post_Code'}, inplace=True)
IPData.rename(columns={'First.IP.Number': 'First_IP_Number'}, inplace=True)
IPData.rename(columns={'Last.IP.Number': 'Last_IP_Number'}, inplace=True)
print('To :',IPData.columns.values)
print('#################################')
#####################################################################
print('#################################')
print('Change ',IPData.columns.values)
for i in IPData.columns.values:
    j='Node_'+i
    IPData.rename(columns={i: j}, inplace=True)
```

```
print('To ', IPData.columns.values)
print('###################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName
print('###################################')
print('Storing :', sFileName)
print('###################################')
IPData.to_csv(sFileName, index = False, encoding="latin-1")
#####################################################################
print('###################################')
print('### Done!! ######################')
print('###################################')
#####################################################################
```

Output:

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

	A	B	C	D	E	F	G
1	Node_Country_Code	Node_Place_Name	Node_Post_Code	Node_Latitude	Node_Longitude	ode_First_IP_Numb	Node_Last_IP_Number
2	BW	Gaborone		-24.6464	25.9119	692781056	692781567
3	BW	Gaborone		-24.6464	25.9119	692781824	692783103
4	BW	Gaborone		-24.6464	25.9119	692909056	692909311
5	BW	Gaborone		-24.6464	25.9119	692909568	692910079
6	BW	Gaborone		-24.6464	25.9119	693051392	693052415
7	BW	Gaborone		-24.6464	25.9119	693078272	693078527
8	BW	Gaborone		-24.6464	25.9119	693608448	693616639
9	BW	Gaborone		-24.6464	25.9119	696929792	696930047
10	BW	Gaborone		-24.6464	25.9119	700438784	700439039
11	BW	Gaborone		-24.6464	25.9119	702075904	702076927
12	BW	Gaborone		-24.6464	25.9119	702498816	702499839
13	BW	Gaborone		-24.6464	25.9119	702516224	702517247
14	BW	Gaborone		-24.6464	25.9119	774162663	774162667
15	BW	Gaborone		-24.6464	25.9119	1401887232	1401887743
16	BW	Gaborone		-24.6464	25.9119	1754209024	1754209279
17	NE	Niamey		13.5167	2.1167	696918528	696919039
18	NE	Niamey		13.5167	2.1167	696922112	696924159
19	NE	Niamey		13.5167	2.1167	701203456	701203711
20	NE	Niamey		13.5167	2.1167	758886912	758887167
21	NE	Niamey		13.5167	2.1167	1347294153	1347294160
22	NE	Niamey		13.5167	2.1167	1755108096	1755108351
23	NE	Niamey		13.5167	2.1167	1755828480	1755828735
24	MZ	Maputo		-25.9653	32.5892	692883456	692883967
25	MZ	Maputo		-25.9653	32.5892	692944896	692946943

C. Write a Python / R program to build directed acyclic graph.
Open your python editor and create a file named Assess-DAG-Location.py in directory C:\VKHCG\01-Vermeulen\02-Assess

Code:

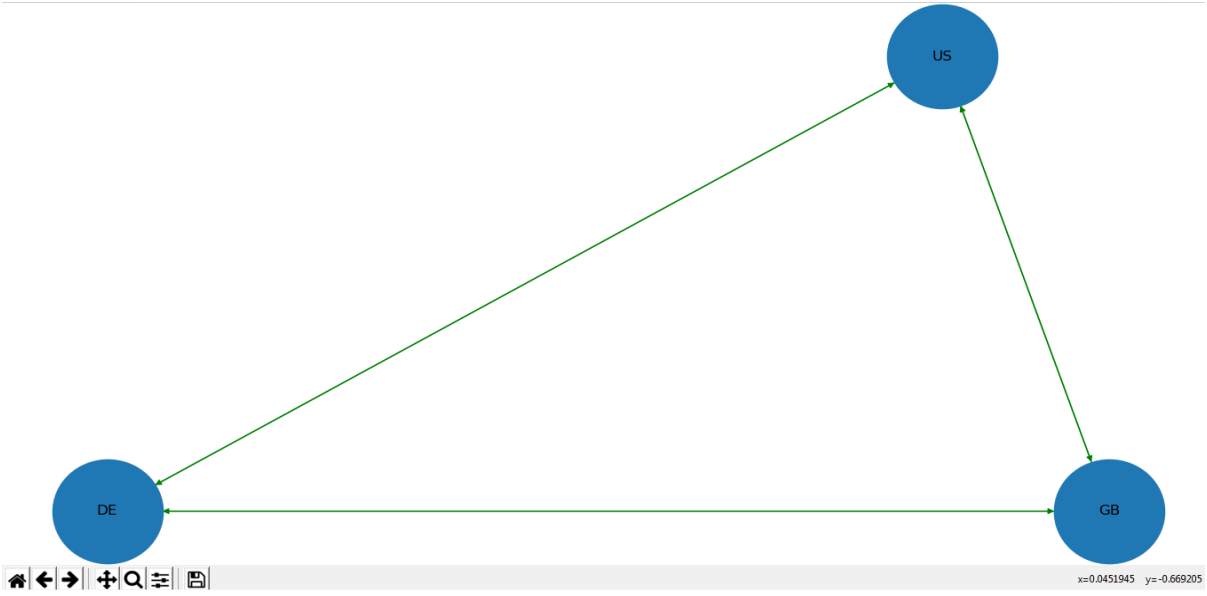
```
################################################################
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName1='Assess-DAG-Company-Country.png'
sOutputFileName2='Assess-DAG-Company-Country-Place.png'
Company='01-Vermeulen'
################################################################
### Import Company Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('##################################')
################################################################
print(CompanyData)
print('##################################')
print('Rows : ',CompanyData.shape[0])
print('##################################')
################################################################
G1=nx.DiGraph()
G2=nx.DiGraph()
################################################################
for i in range(CompanyData.shape[0]):
    G1.add_node(CompanyData['Country'][i])
    sPlaceName= CompanyData['Place_Name'][i] + '-' + CompanyData['Country'][i]
    G2.add_node(sPlaceName)
print('##################################')
for n1 in G1.nodes():
```

```
    for n2 in G1.nodes():
        if n1 != n2:
            print('Link :',n1,' to ', n2)
            G1.add_edge(n1,n2)
print('#################################')
print('#################################')
print("Nodes of graph: ")
print(G1.nodes())
print("Edges of graph: ")
print(G1.edges())
print('#################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName1
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.draw(G1,pos=nx.spectral_layout(G1),
        nodecolor='r',edge_color='g',
        with_labels=True,node_size=8000,
        font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
print('#################################')
for n1 in G2.nodes():
    for n2 in G2.nodes():
        if n1 != n2:
            print('Link :',n1,' to ', n2)
            G2.add_edge(n1,n2)
print('#################################')
print('#################################')
print("Nodes of graph: ")
print(G2.nodes())
print("Edges of graph: ")
print(G2.edges())
print('#################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName2
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.draw(G2,pos=nx.spectral_layout(G2),
```

```
        nodecolor='r',edge_color='b',
         with_labels=True,node_size=8000,
         font_size=12)
plt.savefig(sFileName) # save as png
plt.show()
```

Output:

```
#################################
Rows : 150
#################################
#################################
Link : US to DE
Link : US to GB
Link : DE to US
Link : DE to GB
Link : GB to US
Link : GB to DE
#################################
#################################
Nodes of graph:
['US', 'DE', 'GB']
Edges of graph:
[('US', 'DE'), ('US', 'GB'), ('DE', 'US'), ('DE', 'GB'), ('GB', 'US'), ('GB', 'DE')]
#################################
```



Customer Location DAG

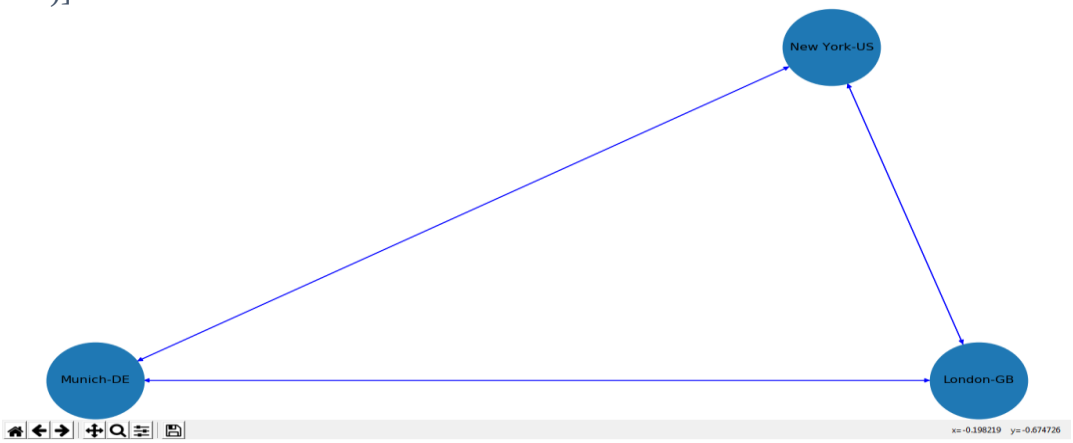
Code:

```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
#####################################################################
Base='C:/VKHCG'
#####################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#####################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName1='Assess-DAG-Company-Country.png'
sOutputFileName2='Assess-DAG-Company-Country-Place.png'
Company='01-Vermeulen'
#####################################################################
### Import Company Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('#################################')
#####################################################################
print(CompanyData)
print('#################################')
print('Rows : ',CompanyData.shape[0])
print('#################################')
#####################################################################
G1=nx.DiGraph()
G2=nx.DiGraph()
#####################################################################
for i in range(CompanyData.shape[0]):
  G1.add_node(CompanyData['Country'][i])
  sPlaceName= CompanyData['Place_Name'][i] + '-' + CompanyData['Country'][i]
  G2.add_node(sPlaceName)
print('#################################')
for n1 in G1.nodes():
  for n2 in G1.nodes():
    if n1 != n2:
      print('Link :',n1,' to ', n2)
    G1.add_edge(n1,n2)
print('#################################')
print('#################################')
print("Nodes of graph: ")
```

```
print(G1.nodes())
print("Edges of graph: ")
print(G1.edges())
print('#################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName1
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.draw(G1,pos=nx.spectral_layout(G1),
  nodecolor='r',edge_color='g',
  with_labels=True,node_size=8000,
  font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
print('#################################')
for n1 in G2.nodes():
  for n2 in G2.nodes():
    if n1 != n2:
      print('Link :',n1,' to ', n2)
     G2.add_edge(n1,n2)
print('#################################')
print('#################################')
print("Nodes of graph: ")
print(G2.nodes())
print("Edges of graph: ")
print(G2.edges())
print('#################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName2
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.draw(G2,pos=nx.spectral_layout(G2),
   nodecolor='r',edge_color='b',
   with_labels=True,node_size=8000,
    font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
```

Output:

```
################################
Link : New York-US to Munich-DE
Link : New York-US to London-GB
Link : Munich-DE to New York-US
Link : Munich-DE to London-GB
Link : London-GB to New York-US
Link : London-GB to Munich-DE
################################
################################
Nodes of graph:
['New York-US', 'Munich-DE', 'London-GB']
Edges of graph:
[('New York-US', 'Munich-DE'), ('New York-US', 'London-GB'), ('Munich-DE', 'New York-US'), ('Munich-DE', 'London-GB'), ('London-GB', 'New York-US'), ('London-GB', 'Munich-DE')]
```



Open your Python editor and create a file named Assess-DAG-GPS.py in directory C:\VKHCG\01-Vermeulen\02-Assess.

Code:

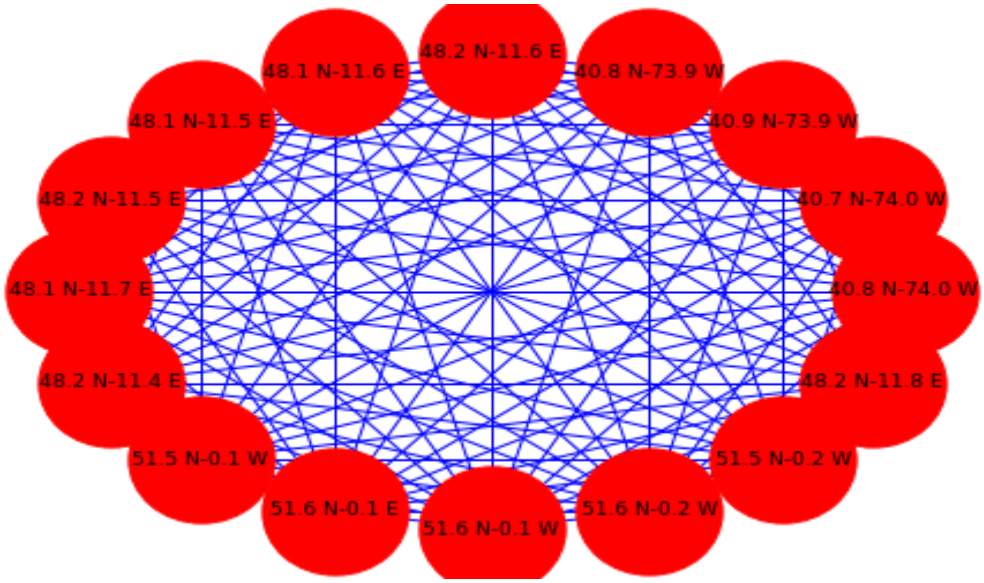
```
########################################################################
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
########################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
########################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName='Assess-DAG-Company-GPS.png'
```

```
Company='01-Vermeulen'
##################################################################
### Import Company Data
##################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('#################################')
##################################################################
print(CompanyData)
print('#################################')
print('Rows : ',CompanyData.shape[0])
print('#################################')
##################################################################
G=nx.Graph()
##################################################################
for i in range(CompanyData.shape[0]):
    nLatitude=round(CompanyData['Latitude'][i],1)
    nLongitude=round(CompanyData['Longitude'][i],1)
    if nLatitude < 0:
        sLatitude = str(nLatitude*-1) + ' S'
    else:
        sLatitude = str(nLatitude) + ' N'
    if nLongitude < 0:
        sLongitude = str(nLongitude*-1) + ' W'
    else:
        sLongitude = str(nLongitude) + ' E'
    sGPS= sLatitude + '-' + sLongitude
    G.add_node(sGPS)
print('#################################')
for n1 in G.nodes():
    for n2 in G.nodes():
        if n1 != n2:
            print('Link :',n1,' to ', n2)
            G.add_edge(n1,n2)
print('#################################')
print('#################################')
print("Nodes of graph: ")
print(G.nodes())
print("Edges of graph: ")
print(G.edges())
print('#################################')
##################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
##################################################################
```

```
sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
pos=nx.circular_layout(G,dim=1, scale=2)
nx.draw(G,pos=pos,
        nodecolor='r',edge_color='b',
        with_labels=True,node_size=4000,
        font_size=9)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
```

Output:

```
################################
Working Base: C:/VKHCG using win32
################################
Loading: C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-
Python/Retrieve_Router_Location.csv
################################
Loaded Company: ['Country' 'Place_Name' 'Latitude' 'Longitude']
################################
Country Place_Name Latitude Longitude
0 US New York 40.7528 -73.9725
1 US New York 40.7214 -74.0052
-
-
-
Link: 48.15 N-11.74 E to 48.15 N-11.46 E
Link: 48.15 N-11.74 E to 48.09 N-11.54 E
Link: 48.15 N-11.74 E to 48.18 N-11.75 E
Link: 48.15 N-11.74 E to 48.1 N-11.47 E
################################
Nodes of graph:
117
Edges of graph:
6786
################################
```



D. Write a Python / R program to pick the content for Bill Boards from the given data.

Code:

```
import sys
mport os
import sqlite3 as sq
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('##################################')
#####################################################################
sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
sOutputFileName='Assess-DE-Billboard-Visitor.csv'
Company='02-Krennwallner'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'
conn = sq.connect(sDatabaseName)
#####################################################################
### Import Billboard Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('##################################')
print('Loading :',sFileName)
```

```
print('#################################')
BillboardRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
BillboardRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BillboardData=BillboardRawData
print('Loaded Company :',BillboardData.columns.values)
print('#################################')
#####################################################################
print('#################')
sTable='Assess_BillboardData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#####################################################################
print(BillboardData.head())
print('#################################')
print('Rows : ',BillboardData.shape[0])
print('#################################')
#####################################################################
### Import Billboard Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('#################################')
print('Loading :',sFileName)
print('#################################')
VisitorRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData[VisitorRawData.Country=='DE']
print('Loaded Company :',VisitorData.columns.values)
print('#################################')
#####################################################################
print('#################')
sTable='Assess_VisitorData'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#####################################################################
print(VisitorData.head())
print('#################################')
print('Rows : ',VisitorData.shape[0])
print('#################################')
#####################################################################
print('#################')
sTable='Assess_BillboardVisitorData'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.Country AS BillboardCountry,"
sSQL=sSQL+ " A.Place_Name AS BillboardPlaceName,"
sSQL=sSQL+ " A.Latitude AS BillboardLatitude, "
sSQL=sSQL+ " A.Longitude AS BillboardLongitude,"
```

```
sSQL=sSQL+ " B.Country AS VisitorCountry,"
sSQL=sSQL+ " B.Place_Name AS VisitorPlaceName,"
sSQL=sSQL+ " B.Latitude AS VisitorLatitude, "
sSQL=sSQL+ " B.Longitude AS VisitorLongitude,"
sSQL=sSQL+ " (B.Last_IP_Number - B.First_IP_Number) * 365.25 * 24 * 12 AS
VisitorYearRate"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_BillboardData as A"
sSQL=sSQL+ " JOIN "
sSQL=sSQL+ " Assess_VisitorData as B"
sSQL=sSQL+ " ON "
sSQL=sSQL+ " A.Country = B.Country"
sSQL=sSQL+ " AND "
sSQL=sSQL+ " A.Place_Name = B.Place_Name;"
BillboardVistorsData=pd.read_sql_query(sSQL, conn)
print('#################')
#####################################################################
print('#################')
sTable='Assess_BillboardVistorsData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardVistorsData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#####################################################################
print(BillboardVistorsData.head())
print('###################################')
print('Rows : ',BillboardVistorsData.shape[0])
print('###################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####################################################################
print('###################################')
print('Storing :', sFileName)
print('###################################')
sFileName=sFileDir + '/' + sOutputFileName
BillboardVistorsData.to_csv(sFileName, index = False)
print('###################################')
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

Output:

C:\VKHCG\02-Krennwallner\01-Retrieve\01-EDS\02-Python\Retrieve_Online_Visitor.csv containing, 10,48,576 (Ten lack Forty Eight Thousand Five Hundred and Seventy Six) rows.

	A	B	C	D	E	F
1	Country	Place_Name	Latitude	Longitude	First_IP_Number	Last_IP_Number
2	BW	Gaborone	-24.6464	25.9119	692781056	692781567
3	BW	Gaborone	-24.6464	25.9119	692781824	692783103
4	BW	Gaborone	-24.6464	25.9119	692909056	692909311
1048556	NL	Amsterdam	52.3556	4.9136	385939968	385940479
1048557	NL	Amsterdam	52.3556	4.9136	385942528	385943551
1048558	NL	Amsterdam	52.3556	4.9136	385957888	385961983
1048559	NL	Amsterdam	52.3556	4.9136	386003200	386003967
1048560	NL	Amsterdam	52.3556	4.9136	386012160	386012671
1048561	NL	Amsterdam	52.3556	4.9136	386013184	386013695
1048562	NL	Amsterdam	52.3556	4.9136	386015232	386015487
1048563	NL	Amsterdam	52.3556	4.9136	386020352	386021375
1048564	NL	Amsterdam	52.3556	4.9136	386035712	386039807
1048565	NL	Amsterdam	52.3556	4.9136	386060288	386068479
1048566	NL	Amsterdam	52.3556	4.9136	386073344	386073599
1048567	NL	Amsterdam	52.3556	4.9136	386074112	386074623
1048568	NL	Amsterdam	52.3556	4.9136	386076416	386076671
1048569	NL	Amsterdam	52.3556	4.9136	386088960	386089983
1048570	NL	Amsterdam	52.3556	4.9136	386095616	386096127
1048571	NL	Amsterdam	52.3556	4.9136	386109440	386113535
1048572	NL	Amsterdam	52.3556	4.9136	386191360	386195455
1048573	NL	Amsterdam	52.3556	4.9136	386201600	386203135
1048574	NL	Amsterdam	52.3556	4.9136	386215936	386220031
1048575	NL	Amsterdam	52.3556	4.9136	386228224	386232319
1048576	NL	Amsterdam	52.3556	4.9136	386244608	386244863

Retrieve_Online_Visitor

SQLite Visitor's Database

C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db Table:
BillboardCountry BillboardPlaceName ... VisitorLongitude VisitorYearRate
0 DE Lake ... 8.5667 26823960.0
1 DE Horb ... 8.6833 26823960.0
2 DE Horb ... 8.6833 53753112.0
3 DE Horb ... 8.6833 107611416.0
4 DE Horb ... 8.6833 13359384.0

	A	B	C	D	E	F	G	H	I
1	illboardCountry	BillboardPlaceName	oardLatit	oardLongi	sitorCount	VisitorPlaceName	sitorLatitu	itorLongitu	VisitorYearRate
2	DE	Lake	51.7833	8.5667	DE	Lake	51.7833	8.5667	26823960
3	DE	Horb	48.4333	8.6833	DE	Horb	48.4333	8.6833	26823960
4	DE	Horb	48.4333	8.6833	DE	Horb	48.4333	8.6833	53753112
5	DE	Horb	48.4333	8.6833	DE	Horb	48.4333	8.6833	107611416
6	DE	Horb	48.4333	8.6833	DE	Horb	48.4333	8.6833	13359384
7	DE	Horb	48.4333	8.6833	DE	Horb	48.4889	8.6734	26823960
8	DE	Horb	48.4333	8.6833	DE	Horb	48.4889	8.6734	53753112
9	DE	Hardenberg	51.1	7.7333	DE	Hardenberg	51.1	7.7333	26823960
181221	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1167	8.6833	1157112
181222	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1167	8.6833	24299352
181223	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1167	8.6833	807769368
181224	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1172	8.7281	53753112
181225	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1172	8.7281	26823960
181226	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1172	8.7281	107611416
181227	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1172	8.7281	1577880
181228	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1184	8.6095	15042456
181229	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1184	8.6095	10834776
181230	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1319	8.6838	736344
181231	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1319	8.6838	0
181232	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1327	8.7668	736344
181233	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1492	8.7097	1723360536
181234	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1492	8.7097	430761240
181235	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1528	8.745	26823960
181236	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1878	8.6632	1577880

Assess-DE-Billboard-Visitor

E. Write a Python / R program to generate GML file from the given csv file.

Code:

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import networkx as nx
import sys
import os
import sqlite3 as sq
import pandas as pd
from geopy.distance import vincenty
#####################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#####################################################################
Company='02-Krennwallner'
sTable='Assess_BillboardVisitorData'
sOutputFileName='Assess-DE-Billboard-Visitor.gml'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'
conn = sq.connect(sDatabaseName)
#####################################################################
print('#################')
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select "
sSQL=sSQL+ " A.BillboardCountry,"
sSQL=sSQL+ " A.BillboardPlaceName,"
sSQL=sSQL+ " ROUND(A.BillboardLatitude,3) AS BillboardLatitude, "
sSQL=sSQL+ " ROUND(A.BillboardLongitude,3) AS BillboardLongitude,"
sSQL=sSQL+ " (CASE WHEN A.BillboardLatitude < 0 THEN "
sSQL=sSQL+ "  'S' || ROUND(ABS(A.BillboardLatitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'N' || ROUND(ABS(A.BillboardLatitude),3)"
sSQL=sSQL+ " END ) AS sBillboardLatitude,"
sSQL=sSQL+ " (CASE WHEN A.BillboardLongitude < 0 THEN "
sSQL=sSQL+ "  'W' || ROUND(ABS(A.BillboardLongitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'E' || ROUND(ABS(A.BillboardLongitude),3)"
sSQL=sSQL+ " END ) AS sBillboardLongitude,"
```

```
sSQL=sSQL+ " A.VisitorCountry,"
sSQL=sSQL+ " A.VisitorPlaceName,"
sSQL=sSQL+ " ROUND(A.VisitorLatitude,3) AS VisitorLatitude, "
sSQL=sSQL+ " ROUND(A.VisitorLongitude,3) AS VisitorLongitude,"
sSQL=sSQL+ " (CASE WHEN A.VisitorLatitude < 0 THEN "
sSQL=sSQL+ "  'S' || ROUND(ABS(A.VisitorLatitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'N' ||ROUND(ABS(A.VisitorLatitude),3)"
sSQL=sSQL+ " END ) AS sVisitorLatitude,"
sSQL=sSQL+ " (CASE WHEN A.VisitorLongitude < 0 THEN "
sSQL=sSQL+ "  'W' || ROUND(ABS(A.VisitorLongitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'E' || ROUND(ABS(A.VisitorLongitude),3)"
sSQL=sSQL+ " END ) AS sVisitorLongitude,"
sSQL=sSQL+ " A.VisitorYearRate"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_BillboardVistorsData AS A;"
BillboardVistorsData=pd.read_sql_query(sSQL, conn)
print('#################')
#####################################################################
BillboardVistorsData['Distance']=BillboardVistorsData.apply(lambda row:
   round(
   vincenty((row['BillboardLatitude'],row['BillboardLongitude']),
             (row['VisitorLatitude'],row['VisitorLongitude'])).miles
         ,4)
     ,axis=1)
#####################################################################
G=nx.Graph()
#####################################################################

for i in range(BillboardVistorsData.shape[0]):
   sNode0='MediaHub-' + BillboardVistorsData['BillboardCountry'][i]

   sNode1='B-'+ BillboardVistorsData['sBillboardLatitude'][i] + '-'
   sNode1=sNode1 + BillboardVistorsData['sBillboardLongitude'][i]
   G.add_node(sNode1,
            Nodetype='Billboard',
            Country=BillboardVistorsData['BillboardCountry'][i],
            PlaceName=BillboardVistorsData['BillboardPlaceName'][i],
            Latitude=round(BillboardVistorsData['BillboardLatitude'][i],3),
            Longitude=round(BillboardVistorsData['BillboardLongitude'][i],3)
   sNode2='M-'+ BillboardVistorsData['sVisitorLatitude'][i] + '-'
   sNode2=sNode2 + BillboardVistorsData['sVisitorLongitude'][i]
   G.add_node(sNode2,
            Nodetype='Mobile',
            Country=BillboardVistorsData['VisitorCountry'][i],
            PlaceName=BillboardVistorsData['VisitorPlaceName'][i],
            Latitude=round(BillboardVistorsData['VisitorLatitude'][i],3),
            Longitude=round(BillboardVistorsData['VisitorLongitude'][i],3))
   print('Link Media Hub :',sNode0,' to Billboard : ', sNode1)
```

```
    G.add_edge(sNode0,sNode1)
    print('Link Post Code :',sNode1,' to GPS : ', sNode2)
    G.add_edge(sNode1,sNode2,distance=round(BillboardVistorsData['Distance'][i]))

#################################################################
print('###################################')
print("Nodes of graph: ",nx.number_of_nodes(G))
print("Edges of graph: ",nx.number_of_edges(G))
print('###################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
sFileName=sFileDir + '/' + sOutputFileName
print('###################################')
print('Storing :', sFileName)
print('###################################')
nx.write_gml(G,sFileName)
sFileName=sFileName +'.gz'
nx.write_gml(G,sFileName)
#################################################################
#################################################################
print('### Done!! ############################################')
#################################################################
```

Output:

This will produce a set of demonstrated values onscreen, plus a graph data file named Assess-DE-Billboard-Visitor.gml.

(It takes a long time to complete the process, after completion the gml file can be viewed in text editor)

Planning an Event for Top-Ten Customers

Open Python editor and create a file named Assess-Visitors.py in directory C:\VKHCG\02-Krennwallner\02-Assess

Code:

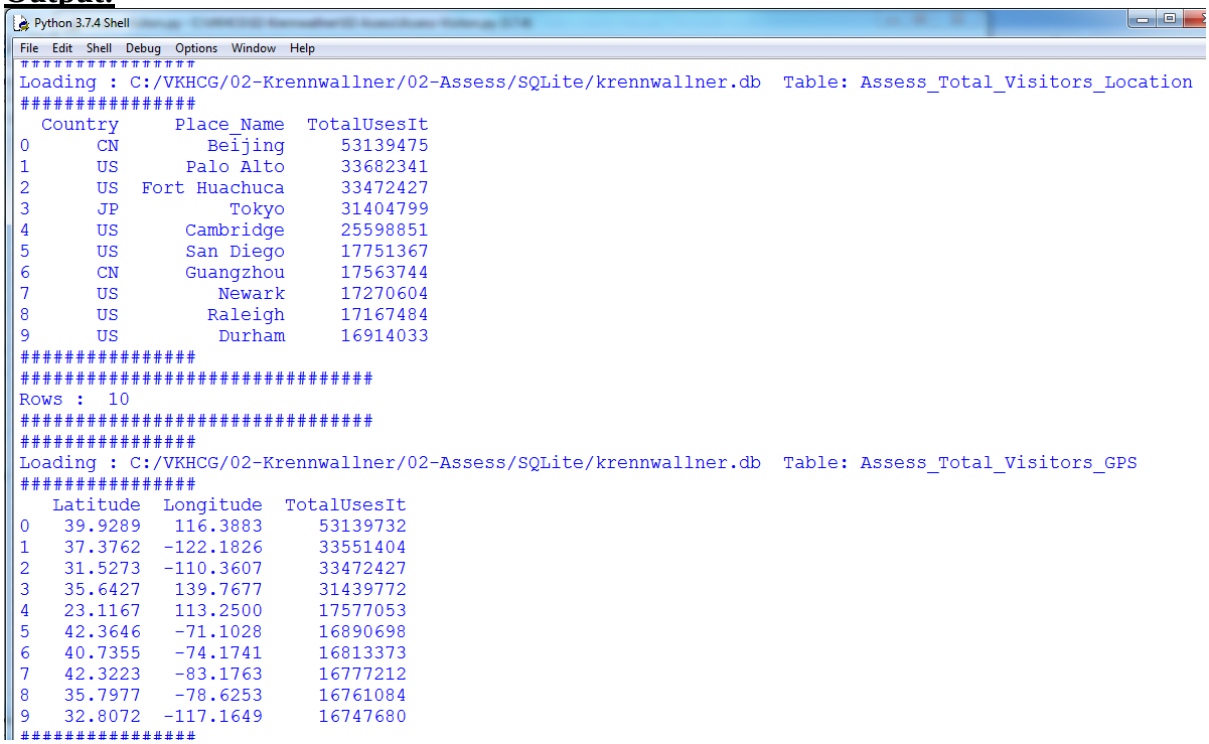
```
#################################################################
import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
```

```
print('################################')
#####################################################################
Company='02-Krennwallner'
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'
conn = sq.connect(sDatabaseName)
#####################################################################
### Import Country Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
VisitorRawData=pd.read_csv(sFileName,
                    header=0,
                    low_memory=False,
                    encoding="latin-1",
                    skip_blank_lines=True)
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData
print('Loaded Company :',VisitorData.columns.values)
print('################################')
#####################################################################
print('################')
sTable='Assess_Visitor'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print('################')
#####################################################################
print(VisitorData.head())
print('################################')
print('Rows : ',VisitorData.shape[0])
print('################################')
#####################################################################
print('################')
sView='Assess_Visitor_UseIt'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " A.Country,"
sSQL=sSQL+ " A.Place_Name,"
sSQL=sSQL+ " A.Latitude,"
sSQL=sSQL+ " A.Longitude,"
```

```
sSQL=sSQL+ " (A.Last_IP_Number - A.First_IP_Number) AS UsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor as A"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " Country is not null"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " Place_Name is not null;"
sql.execute(sSQL,conn)
#####################################################################
print('#################')
sView='Assess_Total_Visitors_Location'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " Country,"
sSQL=sSQL+ " Place_Name,"
sSQL=sSQL+ " SUM(UsesIt) AS TotalUsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor_UseIt"
sSQL=sSQL+ " GROUP BY"
sSQL=sSQL+ " Country,"
sSQL=sSQL+ " Place_Name"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " TotalUsesIt DESC"
sSQL=sSQL+ " LIMIT 10;"
sql.execute(sSQL,conn)
#####################################################################
print('#################')
sView='Assess_Total_Visitors_GPS'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " Latitude,"
sSQL=sSQL+ " Longitude,"
sSQL=sSQL+ " SUM(UsesIt) AS TotalUsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor_UseIt"
sSQL=sSQL+ " GROUP BY"
sSQL=sSQL+ " Latitude,"
sSQL=sSQL+ " Longitude"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " TotalUsesIt DESC"
sSQL=sSQL+ " LIMIT 10;"
sql.execute(sSQL,conn)
#####################################################################
sTables=['Assess_Total_Visitors_Location', 'Assess_Total_Visitors_GPS']
```

```
for sTable in sTables:
    print('#################')
    print('Loading :',sDatabaseName,' Table:',sTable)
    sSQL=" SELECT "
    sSQL=sSQL+ " *"
    sSQL=sSQL+ " FROM"
    sSQL=sSQL+ " " + sTable + ";"
    TopData=pd.read_sql_query(sSQL, conn)
    print('#################')
    print(TopData)
    print('#################')
    print('#################################')
    print('Rows : ',TopData.shape[0])
    print('#################################')
#####################################################################
print('### Done!! ##############################################')
#####################################################################
```

Output:



```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
#################
Loading : C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db  Table: Assess_Total_Visitors_Location
#################
  Country     Place_Name  TotalUsesIt
0      CN        Beijing     53139475
1      US      Palo Alto     33682341
2      US  Fort Huachuca     33472427
3      JP          Tokyo     31404799
4      US      Cambridge     25598851
5      US      San Diego     17751367
6      CN      Guangzhou     17563744
7      US         Newark     17270604
8      US        Raleigh     17167484
9      US         Durham     16914033
#################
################################
Rows :  10
################################
#################
Loading : C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db  Table: Assess_Total_Visitors_GPS
#################
   Latitude  Longitude  TotalUsesIt
0   39.9289   116.3883     53139732
1   37.3762  -122.1826     33551404
2   31.5273  -110.3607     33472427
3   35.6427   139.7677     31439772
4   23.1167   113.2500     17577053
5   42.3646   -71.1028     16890698
6   40.7355   -74.1741     16813373
7   42.3223   -83.1763     16777212
8   35.7977   -78.6253     16761084
9   32.8072  -117.1649     16747680
#################
```

F. Write a Python / R program to plan the locations of the warehouses from the given data

Open your editor and create a file named Assess-Warehouse-Address.py in directory C:\VKHCG\03-Hillman\02-Assess.

Code:

```
import os
import pandas as pd
from geopy.geocoders import Nominatim
geolocator = Nominatim()
################################################################
InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_GB_Postcode_Warehouse.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_GB_Warehouse_Address.csv'
Company='03-Hillman'
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 + '/' + EDSDir
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('############')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False)
Warehouse.sort_values(by='postcode', ascending=1)
## Limited to 10 due to service limit on Address Service.
WarehouseGoodHead=Warehouse[Warehouse.latitude != 0].head(5)
WarehouseGoodTail=Warehouse[Warehouse.latitude != 0].tail(5)
WarehouseGoodHead['Warehouse_Point']=WarehouseGoodHead.apply(lambda row:
          (str(row['latitude'])+','+str(row['longitude']))
          ,axis=1)
WarehouseGoodHead['Warehouse_Address']=WarehouseGoodHead.apply(lambda row:
          geolocator.reverse(row['Warehouse_Point']).address
         ,axis=1)
WarehouseGoodHead.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodHead.drop('id', axis=1, inplace=True)
WarehouseGoodHead.drop('postcode', axis=1, inplace=True)
```

```
WarehouseGoodTail['Warehouse_Point']=WarehouseGoodTail.apply(lambda row:
        (str(row['latitude'])+','+str(row['longitude']))
        ,axis=1)
WarehouseGoodTail['Warehouse_Address']=WarehouseGoodTail.apply(lambda row:
        geolocator.reverse(row['Warehouse_Point']).address
       ,axis=1)
WarehouseGoodTail.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodTail.drop('id', axis=1, inplace=True)
WarehouseGoodTail.drop('postcode', axis=1, inplace=True)
WarehouseGood=WarehouseGoodHead.append(WarehouseGoodTail, ignore_index=True)
print(WarehouseGood)
sFileName=sFileDir + '/' + OutputFileName
WarehouseGood.to_csv(sFileName, index = False)
print('### Done!! ############################################')
```

Output:

```
##############################
Working Base : C:/VKHCG  using Windows
##############################
##########
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_GB_Postcode_Warehouse.csv
    latitude  longitude                                  Warehouse_Address
0  57.135140  -2.117310  35, Broomhill Road, Broomhill, Aberdeen, Aberd...
1  57.138750  -2.090890  South Esplanade West, Torry, Aberdeen, Aberdee...
2  57.101000  -2.110600  A92, Cove and Altens, Aberdeen, Aberdeen City,...
3  57.108010  -2.237760  Colthill Circle, Milltimber, Countesswells, Ab...
4  57.100760  -2.270730  Johnston Gardens East, Peterculter, South Last...
5  53.837717  -1.780013  HM Revenue and Customs, Riverside Estate, Temp...
6  53.794470  -1.766539  Listerhills Road Norcroft Street, Listerhills ...
7  51.518556  -0.714794  Sorting Office, Stafferton Way, Fishery, Maide...
8  54.890923  -2.943847  Royal Mail (Delivery Office), Junction Street,...
9  57.481338  -4.223951  Inverness Sorting & Delivery Office, Strothers...
### Done!! ############################################
>>>
```

G. Write a Python / R program using data science via clustering to determine new warehouses using the given data.

Open Python editor and create a file named Assess-Warehouse-Global.py in directory C:\VKHCG\03-Hillman\02-Assess

Code:

```
######################################################################
# -*- coding: utf-8 -*-
######################################################################
import sys
import os
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='03-Hillman'
InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_All_Countries.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_All_Warehouse.csv'
#####################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#####################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('############')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#####################################################################
sColumns={'X1' : 'Country',
        'X2' : 'PostCode',
        'X3' : 'PlaceName',
        'X4' : 'AreaName',
        'X5' : 'AreaCode',
        'X10' : 'Latitude',
        'X11' : 'Longitude'}
Warehouse.rename(columns=sColumns,inplace=True)
WarehouseGood=Warehouse
#####################################################################
sFileName=sFileDir + '/' + OutputFileName
WarehouseGood.to_csv(sFileName, index = False)
#####################################################################
print('### Done!! ################################################')
#####################################################################
```

Output:

```
>>>
===== RESTART: C:\VKHCG\03-Hillman\02-Assess\Assess-Warehouse-Global.py =====
################################
Working Base : C:/VKHCG  using  win32
################################
############
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_All_Countries.csv
### Done!! ############################################
>>>
```

Ln: 33 Col: 4

Open Assess0_All_Warehose.csv from C:\VKHCG\03-Hillman\02-Assess\01-EDS\02-Python

	A	B	C	D	E	F	G	H
1	Unnamed: 0	Country	PostCode	PlaceName	AreaName	AreaCode	Latitude	Longitude
2	1	AD	AD100	Canillo			42.5833	1.6667
3	2	AD	AD200	Encamp			42.5333	1.6333
4	3	AD	AD300	Ordino			42.6	1.55
5	4	AD	AD400	La Massana			42.5667	1.4833
6	5	AD	AD500	Andorra la Vella			42.5	1.5
31621	31620	AT	4925	Gumpling	OberÃ¶sterreich	4	48.1555	13.4802
31622	31621	AT	4925	Windischhub	OberÃ¶sterreich	4	48.1555	13.4802
31623	31622	AT	4926	Obereselbach	OberÃ¶sterreich	4	48.1917	13.5784
31624	31623	AT	4926	Jetzing	OberÃ¶sterreich	4	48.1555	13.4802
31625	31624	AT	4926	Pilgersham	OberÃ¶sterreich	4	48.1772	13.5855
31626	31625	AT	4926	Grausgrub	OberÃ¶sterreich	4	48.1555	13.4802
31627	31626	AT	4926	Marienkirchen am Ha	OberÃ¶sterreich	4	48.1828	13.577
31628	31627	AT	4926	Stocket	OberÃ¶sterreich	4	48.1555	13.4802
31629	31628	AT	4926	Baching	OberÃ¶sterreich	4	48.1555	13.4802
31630	31629	AT	4926	Kern	OberÃ¶sterreich	4	48.1555	13.4802
31631	31630	AT	4926	Manaberg	OberÃ¶sterreich	4	48.1555	13.4802
31632	31631	AT	4926	Untereselbach	OberÃ¶sterreich	4	48.1555	13.4802
31633	31632	AT	4926	Hatting	OberÃ¶sterreich	4	48.1555	13.4802
31634	31633	AT	4926	Unering	OberÃ¶sterreich	4	48.1555	13.4802
31635	31634	AT	4926	Kleinbach	OberÃ¶sterreich	4	48.1555	13.4802
31636	31635	AT	4926	Lehen	OberÃ¶sterreich	4	48.1555	13.4802
31637	31636	AT	4926	Hof	OberÃ¶sterreich	4	48.1555	13.4802
31638	31637	AT	4931	GroÃŸweiffendorf	OberÃ¶sterreich	4	48.15	13.3333
31639	31638	AT	4931	Neulendt	OberÃ¶sterreich	4	48.1697	13.3531

Assess_All_Warehouse

H. Using the given data, write a Python / R program to plan the shipping routes for best-fit international logistics.

Open Python editor and create a file named Assess-Best-Fit-Logistics.py in directory C:\VKHCG\03-Hillman\02-Assess

Code:

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import networkx as nx
from geopy.distance import vincenty
import sqlite3 as sq
from pandas.io import sql
################################################################
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='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_All_Countries.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_Best_Logistics.gml'
#################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn = sq.connect(sDatabaseName)
#################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('############')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#################################################################
sColumns={'X1' : 'Country',
        'X2' : 'PostCode',
        'X3' : 'PlaceName',
        'X4' : 'AreaName',
        'X5' : 'AreaCode',
        'X10' : 'Latitude',
        'X11' : 'Longitude'}
Warehouse.rename(columns=sColumns,inplace=True)
WarehouseGood=Warehouse
#print(WarehouseGood.head())
#################################################################
RoutePointsCountry=pd.DataFrame(WarehouseGood.groupby(['Country'])[['Latitude','Longit
ude']].mean())
#print(RoutePointsCountry.head())
print('#################')
sTable='Assess_RoutePointsCountry'
print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsCountry.to_sql(sTable, conn, if_exists="replace")
print('#################')
#################################################################
RoutePointsPostCode=pd.DataFrame(WarehouseGood.groupby(['Country',
'PostCode'])[['Latitude','Longitude']].mean())
#print(RoutePointsPostCode.head())
```

```
print('################')
sTable='Assess_RoutePointsPostCode'
print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsPostCode.to_sql(sTable, conn, if_exists="replace")
print('################')
#####################################################################
RoutePointsPlaceName=pd.DataFrame(WarehouseGood.groupby(['Country',
'PostCode','PlaceName'])[['Latitude','Longitude']].mean())
#print(RoutePointsPlaceName.head())
print('################')
sTable='Assess_RoutePointsPlaceName'
print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsPlaceName.to_sql(sTable, conn, if_exists="replace")
print('################')
#####################################################################
### Fit Country to Country
#####################################################################
print('################')
sView='Assess_RouteCountries'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry,"
sSQL=sSQL+ "   S.Latitude AS SourceLatitude,"
sSQL=sSQL+ "   S.Longitude AS SourceLongitude,"
sSQL=sSQL+ "   T.Country AS TargetCountry,"
sSQL=sSQL+ "   T.Latitude AS TargetLatitude,"
sSQL=sSQL+ "   T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS S"
sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS T"
sSQL=sSQL+ " WHERE S.Country <> T.Country"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)
print('################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
RouteCountries=pd.read_sql_query(sSQL, conn)
RouteCountries['Distance']=RouteCountries.apply(lambda row:
    round(
    vincenty((row['SourceLatitude'],row['SourceLongitude']),
```

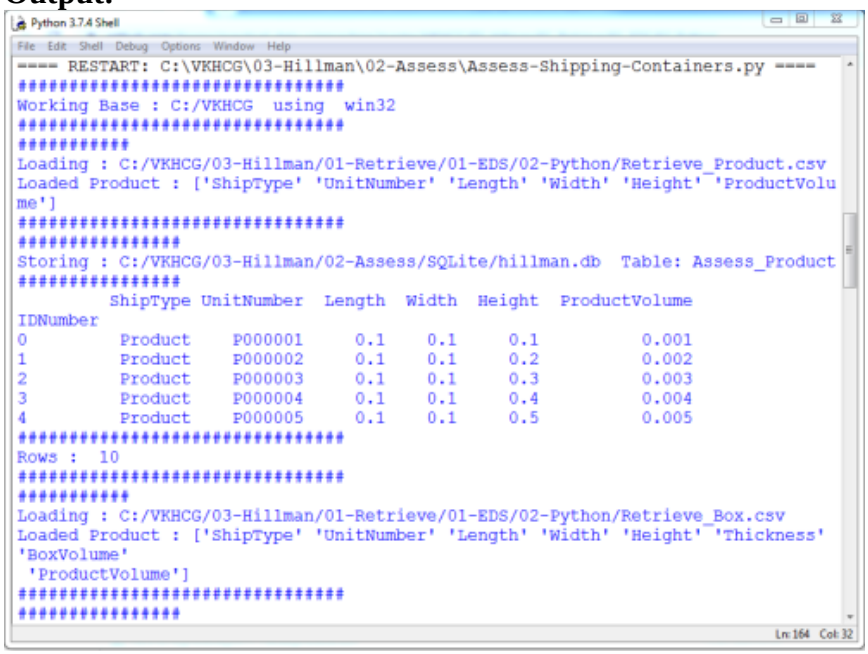
```
            (row['TargetLatitude'],row['TargetLongitude'])).miles
          ,4)
      ,axis=1)
print(RouteCountries.head(5))
#################################################################
### Fit Country to Post Code
#################################################################
print('#################')
sView='Assess_RoutePostCode'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry,"
sSQL=sSQL+ "   S.Latitude AS SourceLatitude,"
sSQL=sSQL+ "   S.Longitude AS SourceLongitude,"
sSQL=sSQL+ "   T.Country AS TargetCountry,"
sSQL=sSQL+ "   T.PostCode AS TargetPostCode,"
sSQL=sSQL+ "   T.Latitude AS TargetLatitude,"
sSQL=sSQL+ "   T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS S"
sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsPostCode AS T"
sSQL=sSQL+ " WHERE S.Country = T.Country"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)
print('#################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
RoutePostCode=pd.read_sql_query(sSQL, conn)
RoutePostCode['Distance']=RoutePostCode.apply(lambda row:
   round(
   vincenty((row['SourceLatitude'],row['SourceLongitude']),
            (row['TargetLatitude'],row['TargetLongitude'])).miles
          ,4)
      ,axis=1)
print(RoutePostCode.head(5))
#################################################################
### Fit Post Code to Place Name
#################################################################
print('#################')
sView='Assess_RoutePlaceName'
```

```
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry,"
sSQL=sSQL+ "   S.PostCode AS SourcePostCode,"
sSQL=sSQL+ "   S.Latitude AS SourceLatitude,"
sSQL=sSQL+ "   S.Longitude AS SourceLongitude,"
sSQL=sSQL+ "   T.Country AS TargetCountry,"
sSQL=sSQL+ "   T.PostCode AS TargetPostCode,"
sSQL=sSQL+ "   T.PlaceName AS TargetPlaceName,"
sSQL=sSQL+ "   T.Latitude AS TargetLatitude,"
sSQL=sSQL+ "   T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsPostCode AS S"
sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsPLaceName AS T"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " S.Country = T.Country"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.PostCode = T.PostCode"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)
print('#################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
RoutePlaceName=pd.read_sql_query(sSQL, conn)
RoutePlaceName['Distance']=RoutePlaceName.apply(lambda row:
   round(
   vincenty((row['SourceLatitude'],row['SourceLongitude']),
            (row['TargetLatitude'],row['TargetLongitude'])).miles
         ,4)
     ,axis=1)
print(RoutePlaceName.head(5))
#####################################################################
G=nx.Graph()
#####################################################################
print('Countries:',RouteCountries.shape)
for i in range(RouteCountries.shape[0]):
   sNode0='C-' + RouteCountries['SourceCountry'][i]
   G.add_node(sNode0,
           Nodetype='Country',
           Country=RouteCountries['SourceCountry'][i],
```

```
            Latitude=round(RouteCountries['SourceLatitude'][i],4),
            Longitude=round(RouteCountries['SourceLongitude'][i],4))
    sNode1='C-' + RouteCountries['TargetCountry'][i]
    G.add_node(sNode1,
            Nodetype='Country',
            Country=RouteCountries['TargetCountry'][i],
            Latitude=round(RouteCountries['TargetLatitude'][i],4),
            Longitude=round(RouteCountries['TargetLongitude'][i],4))
    G.add_edge(sNode0,sNode1,distance=round(RouteCountries['Distance'][i],3))
    #print(sNode0,sNode1)
#####################################################################
print('Post Code:',RoutePostCode.shape)
for i in range(RoutePostCode.shape[0]):
    sNode0='C-' + RoutePostCode['SourceCountry'][i]
    G.add_node(sNode0,
            Nodetype='Country',
            Country=RoutePostCode['SourceCountry'][i],
            Latitude=round(RoutePostCode['SourceLatitude'][i],4),
            Longitude=round(RoutePostCode['SourceLongitude'][i],4))
    sNode1='P-' + RoutePostCode['TargetPostCode'][i]  + '-' +
RoutePostCode['TargetCountry'][i]
    G.add_node(sNode1,
            Nodetype='PostCode',
            Country=RoutePostCode['TargetCountry'][i],
            PostCode=RoutePostCode['TargetPostCode'][i],
            Latitude=round(RoutePostCode['TargetLatitude'][i],4),
            Longitude=round(RoutePostCode['TargetLongitude'][i],4))
    G.add_edge(sNode0,sNode1,distance=round(RoutePostCode['Distance'][i],3))
    #print(sNode0,sNode1)
#####################################################################
print('Place Name:',RoutePlaceName.shape)
for i in range(RoutePlaceName.shape[0]):
    sNode0='P-' + RoutePlaceName['TargetPostCode'][i]  + '-'
    sNode0=sNode0 + RoutePlaceName['TargetCountry'][i]
    G.add_node(sNode0,
            Nodetype='PostCode',
            Country=RoutePlaceName['SourceCountry'][i],
            PostCode=RoutePlaceName['TargetPostCode'][i],
            Latitude=round(RoutePlaceName['SourceLatitude'][i],4),
            Longitude=round(RoutePlaceName['SourceLongitude'][i],4))
    sNode1='L-' + RoutePlaceName['TargetPlaceName'][i]  + '-'
    sNode1=sNode1 + RoutePlaceName['TargetPostCode'][i]  + '-'
    sNode1=sNode1 + RoutePlaceName['TargetCountry'][i]
    G.add_node(sNode1,
            Nodetype='PlaceName',
            Country=RoutePlaceName['TargetCountry'][i],
            PostCode=RoutePlaceName['TargetPostCode'][i],
            PlaceName=RoutePlaceName['TargetPlaceName'][i],
            Latitude=round(RoutePlaceName['TargetLatitude'][i],4),
            Longitude=round(RoutePlaceName['TargetLongitude'][i],4))
```

```
    G.add_edge(sNode0,sNode1,distance=round(RoutePlaceName['Distance'][i],3))
    #print(sNode0,sNode1)
sFileName=sFileDir + '/' + OutputFileName
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.write_gml(G,sFileName)
sFileName=sFileName +'.gz'
nx.write_gml(G,sFileName)
print('#################################')
print('Path:', nx.shortest_path(G,source='P-SW1-GB',target='P-01001-US',weight='distance'))
print('Path length:', nx.shortest_path_length(G,source='P-SW1-GB',target='P-01001-
US',weight='distance'))
print('Path length (1):', nx.shortest_path_length(G,source='P-SW1-GB',target='C-
GB',weight='distance'))
print('Path length (2):', nx.shortest_path_length(G,source='C-GB',target='C-
US',weight='distance'))
print('Path length (3):', nx.shortest_path_length(G,source='C-US',target='P-01001-
US',weight='distance'))
print('#################################')
print('Routes from P-SW1-GB < 2: ', nx.single_source_shortest_path(G,source='P-SW1-GB'
,cutoff=1))
print('Routes from P-01001-US < 2: ', nx.single_source_shortest_path(G,source='P-01001-
US' ,cutoff=1))
print('#################################')
print('#################')
print('Vacuum Database')
sSQL="VACUUM;"
sql.execute(sSQL,conn)
print('#################')
print('### Done!! ############################################')
```

Output:



J. Write a Python program to create a delivery route using the given data.
Open Python editor and create a file named Assess-Shipping-Routes.py in directory C:\VKHCG\03-Hillman\02-Assess.

Code:

```
##################################################################
# -*- coding: utf-8 -*-
##################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import networkx as nx
from geopy.distance import vincenty
##################################################################
nMax=3
nMaxPath=10
nSet=False
nVSet=False
##################################################################
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'
InputDir1='01-Retrieve/01-EDS/01-R'
InputDir2='01-Retrieve/01-EDS/02-Python'
InputFileName1='Retrieve_GB_Postcode_Warehouse.csv'
InputFileName2='Retrieve_GB_Postcodes_Shops.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName1='Assess_Shipping_Routes.gml'
OutputFileName2='Assess_Shipping_Routes.txt'
##################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
##################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
##################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
```

```
    os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/hillman.db'
conn = sq.connect(sDatabaseName)
#################################################################
#################################################################
### Import Warehouse Data
#################################################################
sFileName=Base + '/' + Company + '/' + InputDir1 + '/' + InputFileName1
print('############')
print('Loading :',sFileName)
WarehouseRawData=pd.read_csv(sFileName,
                header=0,
                low_memory=False,
                encoding="latin-1"
                )
WarehouseRawData.drop_duplicates(subset=None, keep='first', inplace=True)
WarehouseRawData.index.name = 'IDNumber'
WarehouseData=WarehouseRawData.head(nMax)
WarehouseData=WarehouseData.append(WarehouseRawData.tail(nMax))
WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode
=='KA13'])
if nSet==True:
WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode
=='SW1W'])
WarehouseData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Loaded Warehouses :',WarehouseData.columns.values)
print('#################################')
#################################################################
print('################')
sTable='Assess_Warehouse_UK'
print('Storing :',sDatabaseName,' Table:',sTable)
WarehouseData.to_sql(sTable, conn, if_exists="replace")
print('################')
#################################################################
print(WarehouseData.head())
print('#################################')
print('Rows : ',WarehouseData.shape[0])
print('#################################')
#################################################################
### Import Shop Data
#################################################################
sFileName=Base + '/' + Company + '/' + InputDir1 + '/' + InputFileName2
print('############')
print('Loading :',sFileName)
ShopRawData=pd.read_csv(sFileName,
                header=0,
                low_memory=False,
                encoding="latin-1"
                )
```

```
ShopRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ShopRawData.index.name = 'IDNumber'
ShopData=ShopRawData
print('Loaded Shops :',ShopData.columns.values)
print('#################################')
#################################################################
print('#################')
sTable='Assess_Shop_UK'
print('Storing :',sDatabaseName,' Table:',sTable)
ShopData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#################################################################
print(ShopData.head())
print('#################################')
print('Rows : ',ShopData.shape[0])
print('#################################')
#################################################################
### Connect HQ
#################################################################
print('#################')
sView='Assess_HQ'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " W.postcode AS HQ_PostCode,"
sSQL=sSQL+ " 'HQ-' || W.postcode AS HQ_Name,"
sSQL=sSQL+ " round(W.latitude,6) AS HQ_Latitude,"
sSQL=sSQL+ " round(W.longitude,6) AS HQ_Longitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Warehouse_UK as W"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " TRIM(W.postcode) in ('KA13','SW1W');"
sql.execute(sSQL,conn)
#################################################################
### Connect Warehouses
#################################################################
print('#################')
sView='Assess_Warehouse'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " W.postcode AS Warehouse_PostCode,"
sSQL=sSQL+ " 'WH-' || W.postcode AS Warehouse_Name,"
sSQL=sSQL+ " round(W.latitude,6) AS Warehouse_Latitude,"
sSQL=sSQL+ " round(W.longitude,6) AS Warehouse_Longitude"
sSQL=sSQL+ " FROM"
```

```
sSQL=sSQL+ " Assess_Warehouse_UK as W;"
sql.execute(sSQL,conn)
#####################################################################
### Connect Warehouse to Shops by PostCode
#####################################################################
print('#################')
sView='Assess_Shop'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " TRIM(S.postcode) AS Shop_PostCode,"
sSQL=sSQL+ " 'SP-' || TRIM(S.FirstCode) || '-' || TRIM(S.SecondCode) AS Shop_Name,"
sSQL=sSQL+ " TRIM(S.FirstCode) AS Warehouse_PostCode,"
sSQL=sSQL+ " round(S.latitude,6) AS Shop_Latitude,"
sSQL=sSQL+ " round(S.longitude,6) AS Shop_Longitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Warehouse_UK as W"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " Assess_Shop_UK as S"
sSQL=sSQL+ " ON"
sSQL=sSQL+ " TRIM(W.postcode) = TRIM(S.FirstCode);"
sql.execute(sSQL,conn)
#####################################################################
#####################################################################
G=nx.Graph()
#####################################################################
print('#################')
sTable = 'Assess_HQ'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData=pd.read_sql_query(sSQL, conn)
print('#################')
#####################################################################
print(RouteData.head())
print('#################################')
print('HQ Rows : ',RouteData.shape[0])
print('#################################')
#####################################################################
for i in range(RouteData.shape[0]):
    sNode0=RouteData['HQ_Name'][i]
    G.add_node(sNode0,
               Nodetype='HQ',
               PostCode=RouteData['HQ_PostCode'][i],
               Latitude=round(RouteData['HQ_Latitude'][i],6),
               Longitude=round(RouteData['HQ_Longitude'][i],6))
```

```
#####################################################################
print('#################')
sTable = 'Assess_Warehouse'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData=pd.read_sql_query(sSQL, conn)
print('#################')
#####################################################################
print(RouteData.head())
print('#################################')
print('Warehouse Rows : ',RouteData.shape[0])
print('#################################')
#####################################################################
for i in range(RouteData.shape[0]):
    sNode0=RouteData['Warehouse_Name'][i]
    G.add_node(sNode0,
            Nodetype='Warehouse',
            PostCode=RouteData['Warehouse_PostCode'][i],
            Latitude=round(RouteData['Warehouse_Latitude'][i],6),
            Longitude=round(RouteData['Warehouse_Longitude'][i],6))
#####################################################################
print('#################')
sTable = 'Assess_Shop'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData=pd.read_sql_query(sSQL, conn)
print('#################')
#####################################################################
print(RouteData.head())
print('#################################')
print('Shop Rows : ',RouteData.shape[0])
print('#################################')
#####################################################################
for i in range(RouteData.shape[0]):
    sNode0=RouteData['Shop_Name'][i]
    G.add_node(sNode0,
            Nodetype='Shop',
            PostCode=RouteData['Shop_PostCode'][i],
            WarehousePostCode=RouteData['Warehouse_PostCode'][i],
            Latitude=round(RouteData['Shop_Latitude'][i],6),
            Longitude=round(RouteData['Shop_Longitude'][i],6))
#####################################################################
## Create Edges
#####################################################################
```

```
print('#################################')
print('Loading Edges')
print('#################################')
for sNode0 in nx.nodes_iter(G):
    for sNode1 in nx.nodes_iter(G):
        if G.node[sNode0]['Nodetype']=='HQ' and \
            G.node[sNode1]['Nodetype']=='HQ' and \
            sNode0 != sNode1:
                distancemeters=round(\
                            vincenty(\
                                    (\
                                     G.node[sNode0]['Latitude'],\
                                     G.node[sNode0]['Longitude']\
                                     ),\
                                     (\
                                      G.node[sNode1]['Latitude']\
                                      ,\
                                      G.node[sNode1]['Longitude']\
                                      )\
                                      ).meters\
                            ,0)
                distancemiles=round(\
                            vincenty(\
                                    (\
                                     G.node[sNode0]['Latitude'],\
                                     G.node[sNode0]['Longitude']\
                                     ),\
                                     (\
                                      G.node[sNode1]['Latitude']\
                                      ,\
                                      G.node[sNode1]['Longitude']\
                                      )\
                                      ).miles\
                            ,3)

                if distancemiles >= 0.05:
                    cost = round(150+(distancemiles * 2.5),6)
                    vehicle='V001'
                else:
                    cost = round(2+(distancemiles * 0.10),6)
                    vehicle='ForkLift'
                G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                        DistanceMiles=distancemiles, \
                        Cost=cost,Vehicle=vehicle)
                if nVSet==True:
                    print('Edge-H-H:',sNode0,' to ', sNode1, \
                      ' Distance:',distancemeters,'meters',\
                      distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
        if G.node[sNode0]['Nodetype']=='HQ' and \
            G.node[sNode1]['Nodetype']=='Warehouse' and \
```

```
    sNode0 != sNode1:
        distancemeters=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).meters\
                        ,0)
        distancemiles=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).miles\
                        ,3)
        if distancemiles >= 10:
            cost = round(50+(distancemiles * 2),6)
            vehicle='V002'
        else:
            cost = round(5+(distancemiles * 1.5),6)
            vehicle='V003'
        if distancemiles <= 50:
            G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                    DistanceMiles=distancemiles, \
                    Cost=cost,Vehicle=vehicle)
            if nVSet==True:
                print('Edge-H-W:',sNode0,' to ', sNode1, \
                  ' Distance:',distancemeters,'meters',\
                  distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
if nSet==True and \
    G.node[sNode0]['Nodetype']=='Warehouse' and \
    G.node[sNode1]['Nodetype']=='Warehouse' and \
    sNode0 != sNode1:
        distancemeters=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
```

```
                                ),\
                                (\
                                 G.node[sNode1]['Latitude']\
                                 ,\
                                 G.node[sNode1]['Longitude']\
                                 )\
                                 ).meters\
                           ,0)
            distancemiles=round(\
                           vincenty(\
                                  (\
                                   G.node[sNode0]['Latitude'],\
                                   G.node[sNode0]['Longitude']\
                                   ),\
                                   (\
                                    G.node[sNode1]['Latitude']\
                                    ,\
                                    G.node[sNode1]['Longitude']\
                                    )\
                                    ).miles\
                           ,3)
            if distancemiles >= 10:
               cost = round(50+(distancemiles * 1.10),6)
               vehicle='V004'
            else:
               cost = round(5+(distancemiles * 1.05),6)
               vehicle='V005'
            if distancemiles <= 20:
               G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                        DistanceMiles=distancemiles, \
                        Cost=cost,Vehicle=vehicle)
               if nVSet==True:
                  print('Edge-W-W:',sNode0,' to ', sNode1, \
                    ' Distance:',distancemeters,'meters',\
                    distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
      if G.node[sNode0]['Nodetype']=='Warehouse' and \
         G.node[sNode1]['Nodetype']=='Shop' and \
         G.node[sNode0]['PostCode']==G.node[sNode1]['WarehousePostCode'] and \
         sNode0 != sNode1:
            distancemeters=round(\
                           vincenty(\
                                  (\
                                   G.node[sNode0]['Latitude'],\
                                   G.node[sNode0]['Longitude']\
                                   ),\
                                   (\
                                    G.node[sNode1]['Latitude']\
                                    ,\
                                    G.node[sNode1]['Longitude']\
                                    )\
```

```
                                ).meters\
                        ,0)
            distancemiles=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).miles\
                        ,3)
            if distancemiles >= 10:
               cost = round(50+(distancemiles * 1.50),6)
               vehicle='V006'
            else:
               cost = round(5+(distancemiles * 0.75),6)
               vehicle='V007'
            if distancemiles <= 10:
               G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                        DistanceMiles=distancemiles, \
                        Cost=cost,Vehicle=vehicle)
               if nVSet==True:
                  print('Edge-W-S:',sNode0,' to ', sNode1, \
                    ' Distance:',distancemeters,'meters',\
                    distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
      if nSet==True and \
         G.node[sNode0]['Nodetype']=='Shop' and \
         G.node[sNode1]['Nodetype']=='Shop' and \
         G.node[sNode0]['WarehousePostCode']==G.node[sNode1]['WarehousePostCode']
and \
         sNode0 != sNode1:
            distancemeters=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).meters\
                        ,0)
            distancemiles=round(\
                        vincenty(\
```

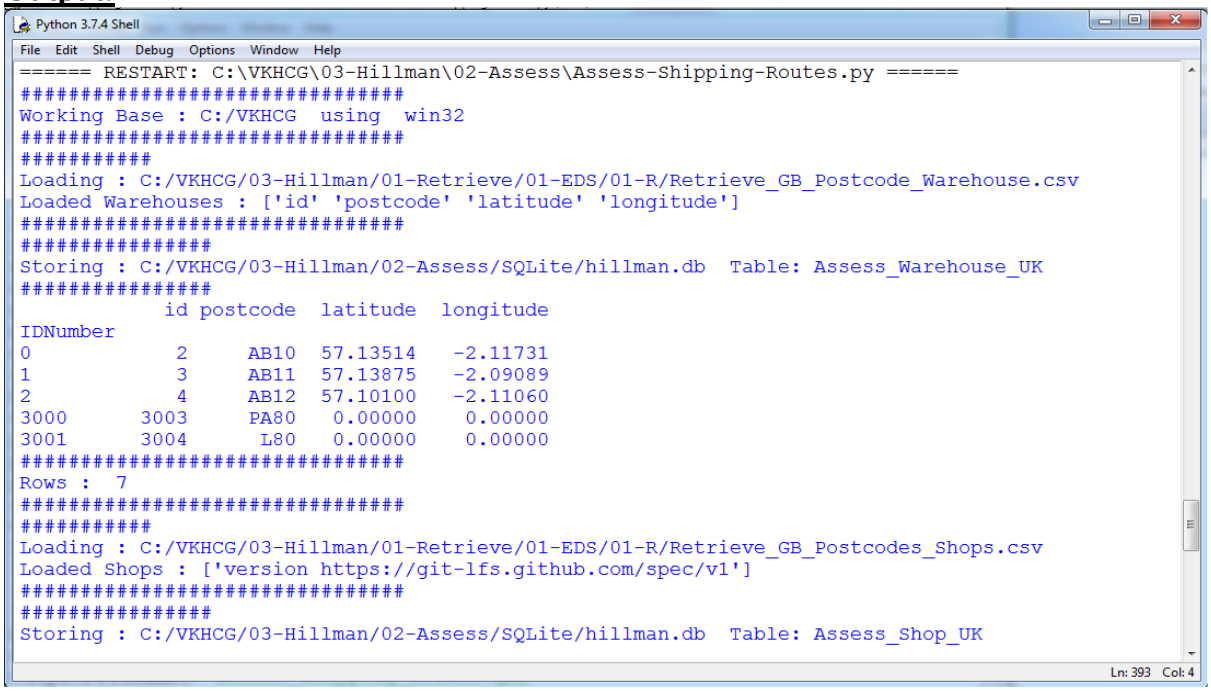
```
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).miles\
                            ,3)
                    if distancemiles >= 0.05:
                        cost = round(5+(distancemiles * 0.5),6)
                        vehicle='V008'
                    else:
                        cost = round(1+(distancemiles * 0.1),6)
                        vehicle='V009'
                    if distancemiles <= 0.075:
                        G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                                DistanceMiles=distancemiles, \
                                Cost=cost,Vehicle=vehicle)
                        if nVSet==True:
                            print('Edge-S-S:',sNode0,' to ', sNode1, \
                              ' Distance:',distancemeters,'meters',\
                              distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

            if nSet==True and \
                G.node[sNode0]['Nodetype']=='Shop' and \
                G.node[sNode1]['Nodetype']=='Shop' and \
                G.node[sNode0]['WarehousePostCode']!=G.node[sNode1]['WarehousePostCode'] and
\
                sNode0 != sNode1:
                    distancemeters=round(\
                                vincenty(\
                                        (\
                                         G.node[sNode0]['Latitude'],\
                                         G.node[sNode0]['Longitude']\
                                         ),\
                                         (\
                                          G.node[sNode1]['Latitude']\
                                          ,\
                                          G.node[sNode1]['Longitude']\
                                          )\
                                          ).meters\
                                ,0)
                    distancemiles=round(\
                                vincenty(\
                                        (\
                                         G.node[sNode0]['Latitude'],\
                                         G.node[sNode0]['Longitude']\
```

```
                                    ),\
                                    (\
                                     G.node[sNode1]['Latitude']\
                                     ,\
                                     G.node[sNode1]['Longitude']\
                                     )\
                                     ).miles\
                                ,3)
                    cost = round(1+(distancemiles * 0.1),6)
                    vehicle='V010'
                    if distancemiles <= 0.025:
                        G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                                DistanceMiles=distancemiles, \
                                Cost=cost,Vehicle=vehicle)
                        if nVSet==True:
                            print('Edge-S-S:',sNode0,' to ', sNode1, \
                             ' Distance:',distancemeters,'meters',\
                             distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
######################################################################
sFileName=sFileDir + '/' + OutputFileName1
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.write_gml(G,sFileName)
sFileName=sFileName +'.gz'
nx.write_gml(G,sFileName)
######################################################################
print('Nodes:',nx.number_of_nodes(G))
print('Edges:',nx.number_of_edges(G))
######################################################################
sFileName=sFileDir + '/' + OutputFileName2
print('#################################')
print('Storing :', sFileName)
print('#################################')
######################################################################
## Create Paths
######################################################################
print('#################################')
print('Loading Paths')
print('#################################')
f = open(sFileName,'w')
l=0
sline = 'ID|Cost|StartAt|EndAt|Path|Measure'
if nVSet==True: print ('0', sline)
f.write(sline+ '\n')
for sNode0 in nx.nodes_iter(G):
    for sNode1 in nx.nodes_iter(G):
        if sNode0 != sNode1 and \
            nx.has_path(G, sNode0, sNode1)==True and \
            nx.shortest_path_length(G, \
```

```
            source=sNode0, \
            target=sNode1, \
            weight='DistanceMiles') < nMaxPath:
              l+=1
              sID='{:.0f}'.format(l)
              spath = ','.join(nx.shortest_path(G, \
                source=sNode0, \
                target=sNode1, \
                weight='DistanceMiles'))
              slength= '{:.6f}'.format(\
                nx.shortest_path_length(G, \
                source=sNode0, \
                target=sNode1, \
                weight='DistanceMiles'))
              sline = sID + '|"DistanceMiles"|"' + sNode0 + '"|"' \
              + sNode1 + '"|"' + spath + '"|' + slength
              if nVSet==True: print (sline)
              f.write(sline + '\n')
              l+=1
              sID='{:.0f}'.format(l)
              spath = ','.join(nx.shortest_path(G, \
                source=sNode0, \
                target=sNode1, \
                weight='DistanceMeters'))
              slength= '{:.6f}'.format(\
                nx.shortest_path_length(G, \
                source=sNode0, \
                target=sNode1, \
                weight='DistanceMeters'))
              sline = sID + '|"DistanceMeters"|"' + sNode0 + '"|"' \
              + sNode1 + '"|"' + spath + '"|' + slength
              if nVSet==True: print (sline)
              f.write(sline + '\n')
              l+=1
              sID='{:.0f}'.format(l)
              spath = ','.join(nx.shortest_path(G, \
                source=sNode0, \
                target=sNode1, \
                weight='Cost'))
              slength= '{:.6f}'.format(\
                nx.shortest_path_length(G, \
                source=sNode0, \
                target=sNode1, \
                weight='Cost'))
              sline = sID + '|"Cost"|"' + sNode0 + '"|"' \
              + sNode1 + '"|"' + spath + '"|' + slength
              if nVSet==True: print (sline)
              f.write(sline + '\n')
f.close()
#####################################################################
```

```
print('Nodes:',nx.number_of_nodes(G))
print('Edges:',nx.number_of_edges(G))
print('Paths:',sID)
#####################################################################
#####################################################################
print('#################')
print('Vacuum Database')
sSQL="VACUUM;"
sql.execute(sSQL,conn)
print('#################')
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

Output:



K. Write a Python program to create Simple forex trading planner from the given data.

Open your Python editor and create a file named Assess-Forex.py in directory C:\VKHCG\04-Clark\02-Assess.

```
#####################################################################
import sys
import os
import sqlite3 as sq
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'
sInputFileName1='01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve-Country-
Currency.csv'
sInputFileName2='04-Clark/01-Retrieve/01-EDS/01-R/Retrieve_Euro_EchangeRates.csv'
#################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
#################################################################
### Import Country Data
#################################################################
sFileName1=Base + '/' + sInputFileName1
print('#################################')
print('Loading :',sFileName1)
print('#################################')
CountryRawData=pd.read_csv(sFileName1,header=0,low_memory=False, encoding="latin-
1")
CountryRawData.drop_duplicates(subset=None, keep='first', inplace=True)
CountryData=CountryRawData
print('Loaded Company :',CountryData.columns.values)
print('#################################')
#################################################################
print('################')
sTable='Assess_Country'
print('Storing :',sDatabaseName,' Table:',sTable)
CountryData.to_sql(sTable, conn, if_exists="replace")
print('################')
#################################################################
print(CountryData.head())
print('#################################')
print('Rows : ',CountryData.shape[0])
print('#################################')
#################################################################
### Import Forex Data
#################################################################
sFileName2=Base + '/' + sInputFileName2
print('#################################')
print('Loading :',sFileName2)
print('#################################')
ForexRawData=pd.read_csv(sFileName2,header=0,low_memory=False, encoding="latin-1")
ForexRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ForexData=ForexRawData.head(5)
print('Loaded Company :',ForexData.columns.values)
print('#################################')
#################################################################
print('################')
```

```
sTable='Assess_Forex'
print('Storing :',sDatabaseName,' Table:',sTable)
ForexData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#####################################################################
print(ForexData.head())
print('#################################')
print('Rows : ',ForexData.shape[0])
print('#################################')
#####################################################################
print('#################')
sTable='Assess_Forex'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.CodeIn"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_Forex as A;"
CodeData=pd.read_sql_query(sSQL, conn)
print('#################')
#####################################################################

for c in range(CodeData.shape[0]):
    print('#################')
    sTable='Assess_Forex & 2x Country > ' + CodeData['CodeIn'][c]
    print('Loading :',sDatabaseName,' Table:',sTable)
    sSQL="select distinct"
    sSQL=sSQL+ " A.Date,"
    sSQL=sSQL+ " A.CodeIn,"
    sSQL=sSQL+ " B.Country as CountryIn,"
    sSQL=sSQL+ " B.Currency as CurrencyNameIn,"
    sSQL=sSQL+ " A.CodeOut,"
    sSQL=sSQL+ " C.Country as CountryOut,"
    sSQL=sSQL+ " C.Currency as CurrencyNameOut,"
    sSQL=sSQL+ " A.Rate"
    sSQL=sSQL+ " from"
    sSQL=sSQL+ " Assess_Forex as A"
    sSQL=sSQL+ " JOIN"
    sSQL=sSQL+ " Assess_Country as B"
    sSQL=sSQL+ " ON A.CodeIn = B.CurrencyCode"
    sSQL=sSQL+ " JOIN"
    sSQL=sSQL+ " Assess_Country as C"
    sSQL=sSQL+ " ON A.CodeOut = C.CurrencyCode"
    sSQL=sSQL+ " WHERE"
    sSQL=sSQL+ " A.CodeIn ='" + CodeData['CodeIn'][c] + "';"
    ForexData=pd.read_sql_query(sSQL, conn).head(1000)
    print('#################')
    print(ForexData)
    print('#################')
    sTable='Assess_Forex_' + CodeData['CodeIn'][c]
    print('Storing :',sDatabaseName,' Table:',sTable)
```

```
    ForexData.to_sql(sTable, conn, if_exists="replace")
    print('################')
    print('################################')
    print('Rows : ',ForexData.shape[0])
    print('################################')
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

Output:

This will produce a set of demonstrated values onscreen by removing duplicate records and other related data processing

L. Write a Python program to process the balance sheet to ensure that only good data is processing.

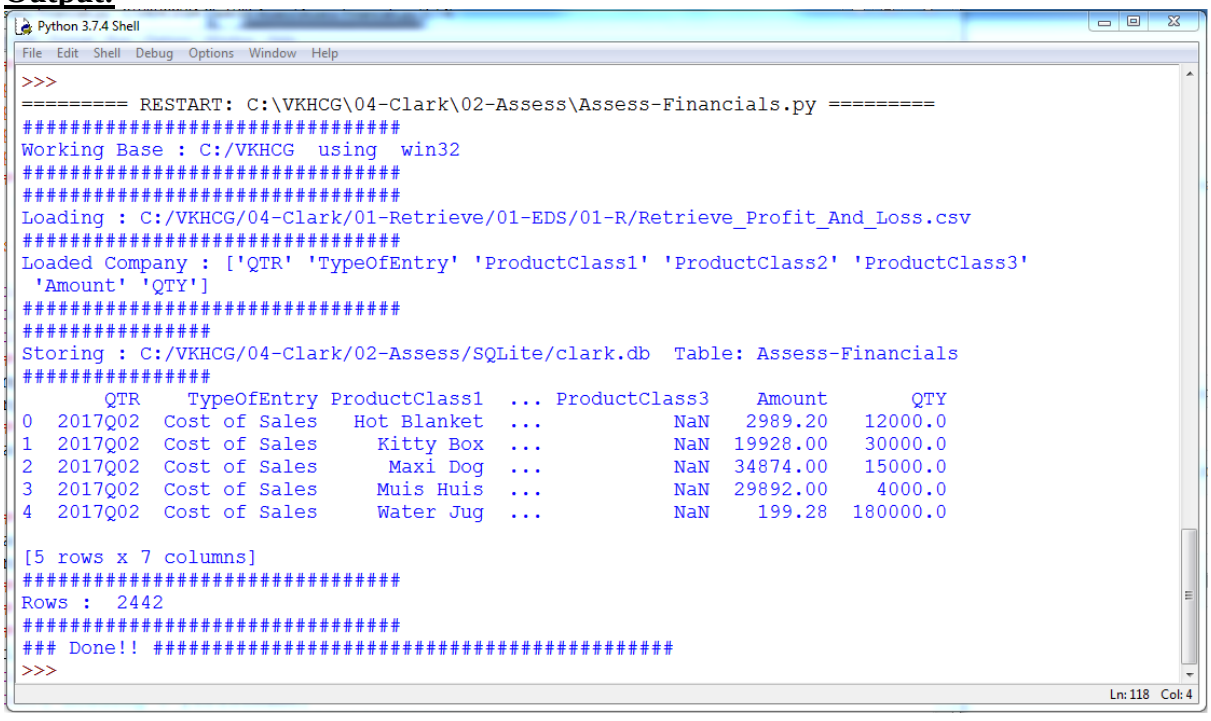
Open Python editor and create a file named Assess-Financials.py in directory C:\VKHCG\04-Clark\02-Assess.

Code:

```
#####################################################################
import sys
import os
import sqlite3 as sq
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='01-Retrieve/01-EDS/01-R/Retrieve_Profit_And_Loss.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
#####################################################################
### Import Financial Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
```

```
FinancialRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
FinancialData=FinancialRawData
print('Loaded Company :',FinancialData.columns.values)
print('###################################')
#####################################################################
print('#################')
sTable='Assess-Financials'
print('Storing :',sDatabaseName,' Table:',sTable)
FinancialData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#####################################################################
print(FinancialData.head())
print('###################################')
print('Rows : ',FinancialData.shape[0])
print('###################################')
#####################################################################
#####################################################################
print('### Done!! ################################################')
#####################################################################
```

Output:



Write a Python program to store all master records for the financial calendar
Open Python editor and create a file named Assess-Calendar.py in directory C:\VKHCG\04-Clark\02-Assess

Code:

```
import sys
import os
import sqlite3 as sq
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'
##################################################################
sDataBaseDirIn=Base + '/' + Company + '/01-Retrieve/SQLite'
if not os.path.exists(sDataBaseDirIn):
    os.makedirs(sDataBaseDirIn)
sDatabaseNameIn=sDataBaseDirIn + '/clark.db'
connIn = sq.connect(sDatabaseNameIn)
##################################################################
sDataBaseDirOut=Base + '/' + Company + '/01-Retrieve/SQLite'
if not os.path.exists(sDataBaseDirOut):
    os.makedirs(sDataBaseDirOut)
sDatabaseNameOut=sDataBaseDirOut + '/clark.db'
connOut = sq.connect(sDatabaseNameOut)
##################################################################
sTableIn='Retrieve_Date'
sSQL='select * FROM ' + sTableIn + ';'
print('#################')
sTableOut='Assess_Time'
print('Loading :',sDatabaseNameIn,' Table:',sTableIn)
dateRawData=pd.read_sql_query(sSQL, connIn)
dateData=dateRawData
##################################################################
print('################################')
print('Load Rows : ',dateRawData.shape[0], ' records')
print('################################')
dateData.drop_duplicates(subset='FinDate', keep='first', inplace=True)
##################################################################
print('#################')
sTableOut='Assess_Date'
print('Storing :',sDatabaseNameOut,' Table:',sTableOut)
dateData.to_sql(sTableOut, connOut, if_exists="replace")
print('#################')
```

```
#####################################################################
print('##################################')
print('Store Rows : ',dateData.shape[0], ' records')
print('##################################')
#####################################################################
#####################################################################
sTableIn='Retrieve_Time'
sSQL='select * FROM ' + sTableIn + ';'
print('#################')
sTableOut='Assess_Time'
print('Loading :',sDatabaseNameIn,' Table:',sTableIn)
timeRawData=pd.read_sql_query(sSQL, connIn)
timeData=timeRawData
#####################################################################
print('##################################')
print('Load Rows : ',timeData.shape[0], ' records')
print('##################################')
timeData.drop_duplicates(subset=None, keep='first', inplace=True)
#####################################################################
print('#################')
sTableOut='Assess_Time'
print('Storing :',sDatabaseNameOut,' Table:',sTableOut)
timeData.to_sql(sTableOut, connOut, if_exists="replace")
print('#################')
#####################################################################
print('##################################')
print('Store Rows : ',timeData.shape[0], ' records')
print('##################################')
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

Output:

```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
========== RESTART: C:\VKHCG\04-Clark\02-Assess\Assess-People.py ==========
Loading : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve-Data_female-names.csv
C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve-Data_female-names.csv
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FemaleName
Rows :  100  records
Loading : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve-Data_male-names.csv
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_MaleName
Rows :  100  records
Loading : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve-Data_last-names.csv
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_Surname
Rows :  200  records
Loading : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FemaleName & Assess_MaleName
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FirstName
Loading : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FirstName x2 & Assess_Surname
Loading : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FirstName & Assess_Surname
        FirstName SecondName  LastName  Gender
2471856    Miguel      Efren    Ortega    Male
3466902    Tommye    Coretta   Roberts  Female
3336496      Stan     Xavier     Costa    Male
1151796   Faviola       Gene     Heard  Female
893614     Dorene     Joelle   Mccloud  Female
...           ...        ...       ...     ...
31958    Santiago              Crook    Male
32635     Shaunte            Ferreira  Female
2436       Bernie              Dubose    Male
39951     Zoraida              Cherry  Female
7702       Dannie                Foret    Male

[20000 rows x 4 columns]
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_People
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
################################
```

Practical No. 6

Aim: Processing Data

Step 1:
Make sure you are using the latest version of **Python**.
Step 2:
Install all the following modules before performing the practical:

Commands For the modules:

pip install quandl

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

Code:

```
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.from_items(TimeLine)
    else:
       TimeRow = pd.DataFrame.from_items(TimeLine)
       TimeFrame = TimeFrame.append(TimeRow)
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
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.from_items(TimeZoneLine)
       else:
          TimeZoneRow = pd.DataFrame.from_items(TimeZoneLine)
          TimeZoneFrame = TimeZoneFrame.append(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!! ############################################')
```

Step 3:
Save the above code in Python IDLE editor with .py extension.
Step 4:
Create another .py file

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='D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/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('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('################################')
sTable='Process_Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_data.to_sql(sTable, conn1, if_exists="replace")
print('################')
PersonHubRaw=Data[['PersonID','FirstName','SecondName','LastName','BirthDateKey']]
PersonHubRaw['PersonHubID']=RawData.apply(lambda row:
        str(uuid.uuid4())
        ,axis=1)
PersonHub=PersonHubRaw.drop_duplicates(subset=None, \
                        keep='first',\
                        inplace=False)
indexed_PersonHub = PersonHub.set_index(['PersonHubID'])
sTable = 'Hub-Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonHub.to_sql(sTable, conn2, if_exists="replace")
PersonSatelliteGenderRaw=Data[['PersonID','FirstName','SecondName','LastName'\
```

```
                    ,'BirthDateKey','Gender']]
PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lambda row:
            str(uuid.uuid4())
            ,axis=1)
PersonSatelliteGender=PersonSatelliteGenderRaw.drop_duplicates(subset=None, \
                                            keep='first', \
                                            inplace=False)
indexed_PersonSatelliteGender = PersonSatelliteGender.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Gender'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteGender.to_sql(sTable, conn2, if_exists="replace")
PersonSatelliteBirthdayRaw=Data[['PersonID','FirstName','SecondName','LastName',\
                    'BirthDateKey','TimeZone','BirthDate']]
PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lambda row:
            str(uuid.uuid4())
            ,axis=1)
PersonSatelliteBirthday=PersonSatelliteBirthdayRaw.drop_duplicates(subset=None, \
                                            keep='first',\
                                            inplace=False)
indexed_PersonSatelliteBirthday = PersonSatelliteBirthday.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Names'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteBirthday.to_sql(sTable, 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('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('### Done!! ############################################')
```

Output:

```
 RESTART: D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\04-Clark\03-Process\Process-People.py
################################
Working Base : D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/VKHCG  using  win32
################################
Loading : D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
################################
D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
################################
Storing : D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/VKHCG/88-DV/datavault.db  Table: Process_Person
###############

Warning (from warnings module):
  File "D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\04-Clark\03-Process\Process-People.py", line 87
    ,axis=1)
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
Storing : D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/VKHCG/88-DV/datavault.db  Table: Hub-Person

Warning (from warnings module):
  File "D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\04-Clark\03-Process\Process-People.py", line 100
    ,axis=1)
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
Storing : D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/VKHCG/88-DV/datavault.db  Table: Satellite-Person-Gender
Storing : D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/VKHCG/88-DV/datavault.db  Table: Satellite-Person-Names
################################
Storing : D:/D-Drive/SIES/Msc IT/Msc IT (Part-1)/Practical Data Science/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Satellite-Person-Names.csv
################################
Vacuum Databases
### Done!! ############################################
>>>
```

B. 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']].copy()
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()
################################################################
```

Output:

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

[3885 rows x 2 columns]
################
Vacuum Databases
################
>>>
```

C. 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.from_items(LocationLine)
      else:
         LocationRow = pd.DataFrame.from_items(LocationLine)
         LocationFrame = LocationFrame.append(LocationRow)
################################################################
LocationHubIndex=LocationFrame.set_index(['IDNumber'],inplace=False)
################################################################
sTable = 'Process-Location'
print('Storing :',sDatabaseName,' Table:',sTable)
LocationHubIndex.to_sql(sTable, conn1, if_exists="replace")
```

```
#################################################################
sTable = 'Hub-Location'
print('Storing :',sDatabaseName,' Table:',sTable)
LocationHubIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('#################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#################')
#################################################################
print('### Done!! ############################################')
#################################################################
```

Output:

```
Create: 641  of  64800 : L+170000-+010000
Create: 642  of  64800 : L+170000-+020000
Create: 643  of  64800 : L+170000-+030000
Create: 644  of  64800 : L+170000-+040000
Create: 645  of  64800 : L+170000-+050000
Create: 646  of  64800 : L+170000-+060000
Create: 647  of  64800 : L+170000-+070000
Create: 648  of  64800 : L+170000-+080000
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process-Location
Storing : C:/VKHCG/88-DV/datavault.db  Table: Hub-Location
################
Vacuum Databases
################
### Done!! ############################################
>>>
```

D. 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:

```
>>>
 RESTART: D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\04-Clark\03-Process\Process-Shares-Data.py
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/04-Clark/00-RawData/VKHCG_Shares.csv
################################
Rows    : 10
Columns: 3
################
################################
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_Shares.csv
################################
################################
################################
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_Shares.csv
################################
################################
################################
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_Shares.csv
################################
################################
################
Share   : WIKI/GOOGL
Rows    : 3424
Columns: 14
################
################
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db  Table: WIKI_Google
Warning (from warnings module):
  File "C:\Users\Admin\AppData\Local\Programs\Python\Python37-32\lib\site-packages\pandas\core\generic.py",
    method=method,
UserWarning: The spaces in these column names will not be changed. In pandas versions < 0.14, spaces were co
################
################################
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Google.csv
################################
################################
################################
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Google.csv
################################
################################
################################
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Google.csv
################################
################################
################
Share   : WIKI/MSFT
Rows    : 8076
Columns: 14
################
#################
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db  Table: FED_RXI_N_A_CA
#################
################################
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_FED_RXI_N_A_CA.csv
################################
################################
################################
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_FED_RXI_N_A_CA.csv
################################
################################
################################
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_FED_RXI_N_A_CA.csv
################################
################################
### Done!! ############################################
>>>
```

Practical No. 7

Aim: Transforming Data

Step 1:

Make sure you are using the latest version of **Python**.

Step 2:

Install all the following modules before performing the practical:

Commands For the modules:

```
pip install sklearn
pip install pillow
pip install matplotlib
pip install opencv-python
pip install sqlite3
pip install datetime
```

A. Transform Superstep

Transform-Gunnarsson_is_Born

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.db'
conn1 = sq.connect(sDatabaseName)
######################################################################
```

```
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
   os.makedirs(sDataVaultDir)
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
   os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
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.from_items(TimeLine)
#####################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Hub-Time-Gunnarsson'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
```

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

```
##################################
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 09:15:00 (-01) (-0100)
##################################
Warning (from warnings module):
  File "D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\01-Ve
rmeulen\04-Transform\Transform-Gunnarsson_is_Born.py", line 73
    TimeFrame = pd.DataFrame.from_items(TimeLine)
FutureWarning: from_items is deprecated. Please use DataFrame.from_dict(dict(item
s), ...) instead. DataFrame.from_dict(OrderedDict(items)) may be used to preserve
the key order.

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

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

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

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

##################################
Person Category
Name: Guðmundur Gunnarsson
Birth Date: 1960-12-20 09:15:00
Birth Zone: Atlantic/Reykjavik
UTC Birth Date: 1960-12-20 10:15:00
##################################
Warning (from warnings module):
  File "D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\01-Ve
rmeulen\04-Transform\Transform-Gunnarsson_is_Born.py", line 114
    PersonFrame = pd.DataFrame.from_items(PersonLine)
FutureWarning: from_items is deprecated. Please use DataFrame.from_dict(dict(item
s), ...) instead. DataFrame.from_dict(OrderedDict(items)) may be used to preserve
the key order.

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

##################################
>>>
```

Transform-Gunnarsson-Sun-Model

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'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir):
   os.makedirs(sDataWarehousetDir)
#####################################################################
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
print('\n#################################')
print('Time Dimension')
BirthZone = 'Atlantic/Reykjavik'
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
```

```
#####################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [IDTimeNumber]),
        ('UTCDate', [BirthDateZoneStr]),
        ('LocalTime', [BirthDateLocal]),
        ('TimeZone', [BirthZone])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
#####################################################################
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
#####################################################################
sTable = 'Dim-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('\n#################################')
print('Dimension Person')
print('\n#################################')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
#####################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [IDPersonNumber]),
            ('FirstName', [FirstName]),
            ('LastName', [LastName]),
            ('Zone', ['UTC']),
            ('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
#####################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####################################################################
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('\n#################################')
print('Fact - Person - time')
print('\n#################################')
IDFactNumber=str(uuid.uuid4())
PersonTimeLine=[('IDNumber', [IDFactNumber]),
              ('IDPersonNumber', [IDPersonNumber]),
              ('IDTimeNumber', [IDTimeNumber])]
PersonTimeFrame = pd.DataFrame.from_items(PersonTimeLine)
```

```
#####################################################################
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Fact-Person-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
```

Output:

```
################################
Working Base : C:/VKHCG  using  win32
################################

################################
Time Dimension
Warning (from warnings module):
  File "D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\01-V
ermeulen\04-Transform\Transform-Gunnarsson-Sun-Model.py", line 53
    TimeFrame = pd.DataFrame.from_items(TimeLine)
FutureWarning: from_items is deprecated. Please use DataFrame.from_dict(dict(ite
ms), ...) instead. DataFrame.from_dict(OrderedDict(items)) may be used to preser
ve the key order.

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

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

################################
Dimension Person

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

Warning (from warnings module):
  File "D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\01-V
ermeulen\04-Transform\Transform-Gunnarsson-Sun-Model.py", line 77
    PersonFrame = pd.DataFrame.from_items(PersonLine)
FutureWarning: from_items is deprecated. Please use DataFrame.from_dict(dict(ite
ms), ...) instead. DataFrame.from_dict(OrderedDict(items)) may be used to preser
ve the key order.

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

################################
>>>
```

B. Building a Data Warehouse

Code:

```
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
Company='01-Vermeulen'
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
    os.makedirs(sDataVaultDir)
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
sSQL=" SELECT DateTimeValue FROM [Hub-Time];"
DateDataRaw=pd.read_sql_query(sSQL, conn2)
DateData=DateDataRaw.head(1000)
print(DateData)
print('\n#################################')
print('Time Dimension')
print('\n#################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
    BirthZone = ('Atlantic/Reykjavik','Europe/London','UCT')
    for j in range(len(BirthZone)):
        t+=1
        print(t,mt*3)
        BirthDateUTC = datetime.strptime(DateData['DateTimeValue'][i],"%Y-%m-%d
%H:%M:%S")
```

```
        BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
        BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
        BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
        BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone[j]))
        BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
        BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
        IDTimeNumber=str(uuid.uuid4())
        TimeLine=[('TimeID', [str(IDTimeNumber)]),
                ('UTCDate', [str(BirthDateZoneStr)]),
                ('LocalTime', [str(BirthDateLocal)]),
                ('TimeZone', [str(BirthZone)])]
        if t==1:
            TimeFrame = pd.DataFrame.from_items(TimeLine)
        else:
            TimeRow = pd.DataFrame.from_items(TimeLine)
            TimeFrame=TimeFrame.append(TimeRow)
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
sTable = 'Dim-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn3, if_exists="replace")
sSQL=" SELECT " + \
     " FirstName," + \
     " SecondName," + \
     " LastName," + \
     " BirthDateKey " + \
     " FROM [Hub-Person];"
PersonDataRaw=pd.read_sql_query(sSQL, conn2)
PersonData=PersonDataRaw.head(1000)
print('\n#################################')
print('Dimension Person')
print('\n#################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
    t+=1
    print(t,mt)
    FirstName = str(PersonData["FirstName"])
    SecondName = str(PersonData["SecondName"])
    if len(SecondName) > 0:
        SecondName=""
    LastName = str(PersonData["LastName"])
    BirthDateKey = str(PersonData["BirthDateKey"])
    IDPersonNumber=str(uuid.uuid4())
    PersonLine=[('PersonID', [str(IDPersonNumber)]),
                ('FirstName', [FirstName]),
```

```
            ('SecondName', [SecondName]),
            ('LastName', [LastName]),
            ('Zone', [str('UTC')]),
            ('BirthDate', [BirthDateKey])]
    if t==1:
        PersonFrame = pd.DataFrame.from_items(PersonLine)
    else:
        PersonRow = pd.DataFrame.from_items(PersonLine)
        PersonFrame = PersonFrame.append(PersonRow)
DimPerson=PersonFrame
print(DimPerson)
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
```

Output:

```
994 1000
995 1000
996 1000
997 1000
998 1000
999 1000
1000 1000
                                PersonID  ...
     BirthDate
0   b2139aa1-01c1-4dab-8833-7c830366102f  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
0   fdc6fe39-ef27-4504-bd1b-0c8e10f19300  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
0   bd03110c-9db7-4507-b6b3-dde9574eb463  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
0   be97c4bd-5d8a-473b-b500-2522959fe475  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
0   462a6edf-2f2e-4d9b-a1f7-c2188f4ac16d  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
..                                     ...  ...
           ...
0   f32b6290-5d58-4b15-87ed-00c225a0568b  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
0   470ea30f-250b-4696-aa63-7d7440e3879e  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
0   2e675cad-f7f0-41f4-be66-fddf941c8c39  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
0   08621441-55ab-4463-a141-542761cac44b  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...
0   1deb82d6-c3e5-44f1-9fab-12d486ab9c49  ...  0      1934-10-17 22:00:00\n1
1911-12-21 ...

[1000 rows x 6 columns]

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

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

Output will be continues.

You have successfully performed data vault to data warehouse transformation.

C. Simple Linear Regression

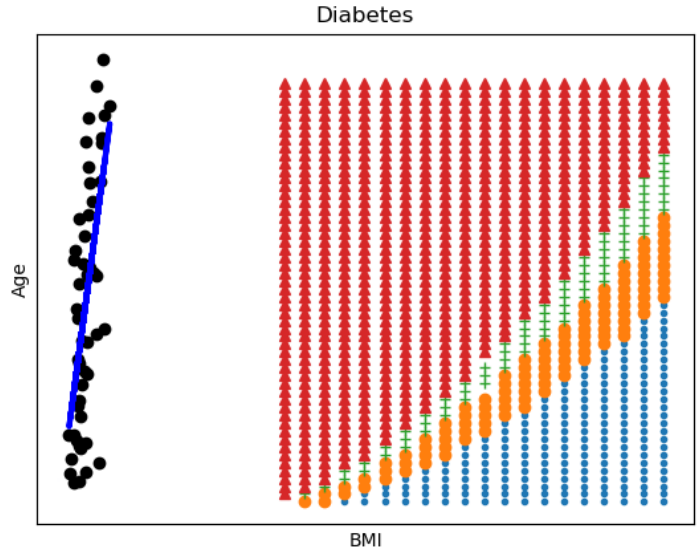
Code:

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
    os.makedirs(sDataVaultDir)
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
#####################################################################
t=0
tMax=((300-100)/10)*((300-30)/5)
for heightSelect in range(100,300,10):
    for weightSelect in range(30,300,5):
        height = round(heightSelect/100,3)
        weight = int(weightSelect)
```

```
        bmi = weight/(height*height)
        if bmi <= 18.5:
            BMI_Result=1
        elif bmi > 18.5 and bmi < 25:
            BMI_Result=2
        elif bmi > 25 and bmi < 30:
            BMI_Result=3
        elif bmi > 30:
            BMI_Result=4
        else:
            BMI_Result=0
        PersonLine=[('PersonID', [str(t)]),
            ('Height', [height]),
            ('Weight', [weight]),
            ('bmi', [bmi]),
            ('Indicator', [BMI_Result])]
        t+=1
        print('Row:',t,'of',tMax)
        if t==1:
            PersonFrame = pd.DataFrame.from_items(PersonLine)
        else:
            PersonRow = pd.DataFrame.from_items(PersonLine)
            PersonFrame = PersonFrame.append(PersonRow)
#####################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####################################################################
sTable = 'Transform-BMI'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
#####################################################################
#####################################################################
sTable = 'Person-Satellite-BMI'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
#####################################################################
sTable = 'Dim-BMI'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
fig = plt.figure()
PlotPerson=DimPerson[DimPerson['Indicator']==1]
x=PlotPerson['Height']
```

```
y=PlotPerson['Weight']
plt.plot(x, y, ".")
PlotPerson=DimPerson[DimPerson['Indicator']==2]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "o")
PlotPerson=DimPerson[DimPerson['Indicator']==3]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "+")
PlotPerson=DimPerson[DimPerson['Indicator']==4]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "^")
plt.axis('tight')
plt.title("BMI Curve")
plt.xlabel("Height(meters)")
plt.ylabel("Weight(kg)")
plt.plot()
# Load the diabetes dataset
diabetes = datasets.load_diabetes()
# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-50:]
diabetes_y_train = diabetes.target[:-30]
diabetes_y_test = diabetes.target[-50:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print('Coefficients: \n', regr.coef_)
print("Mean squared error: %.2f"
% mean_squared_error(diabetes_y_test, diabetes_y_pred))
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
#########################################################################
plt.xticks(())
plt.yticks(())
plt.axis('tight')
plt.title("Diabetes")
plt.xlabel("BMI")
plt.ylabel("Age")
plt.show()
#########################################################################
```

Output:




```
Row: 1064 of 1080.0
Row: 1065 of 1080.0
Row: 1066 of 1080.0
Row: 1067 of 1080.0
Row: 1068 of 1080.0
Row: 1069 of 1080.0
Row: 1070 of 1080.0
Row: 1071 of 1080.0
Row: 1072 of 1080.0
Row: 1073 of 1080.0
Row: 1074 of 1080.0
Row: 1075 of 1080.0
Row: 1076 of 1080.0
Row: 1077 of 1080.0
Row: 1078 of 1080.0
Row: 1079 of 1080.0
Row: 1080 of 1080.0

##################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Transform-BMI

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

##################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Person-Satellite-BMI

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

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

##################################
Coefficients:
 [941.43097333]
Mean squared error: 3477.50
Variance score: 0.41
>>>
```

Practical No. 8

Aim: Organising Data

Step 1:

Make sure you are using the latest version of **Python**.

Step 2:

Install all the following modules before performing the practical:

Commands For the modules:

```
pip install mlxtend
pip install networkx
```

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])
```

Output:

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

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

################################
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Horizontal
################################
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 5
################################
>>>
```

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

Output:

```
################################
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:

```
#####################################################################
# -*- 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)

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:

```
################################
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:

```
#####################################################################
# -*- 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)

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())
```

Output:

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

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

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

E. Association Rule Mining

Code:

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

basket = (df[df['Country'] =="France"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
#####################################################################
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1
#####################################################################
basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)
```

```
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
print(rules.head())

rules[ (rules['lift'] >= 6) &
      (rules['confidence'] >= 0.8) ]
#########################################################################
sProduct1='ALARM CLOCK BAKELIKE GREEN'
print(sProduct1)
print(basket[sProduct1].sum())
sProduct2='ALARM CLOCK BAKELIKE RED'
print(sProduct2)
print(basket[sProduct2].sum())
#########################################################################
basket2 = (df[df['Country'] =="Germany"]
         .groupby(['InvoiceNo', 'Description'])['Quantity']
         .sum().unstack().reset_index().fillna(0)
         .set_index('InvoiceNo'))
basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)

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

Output:

```
################################
Working Base : C:/VKHCG  using  win32
################################
(541909, 8)
                    antecedents  ... conviction
0  (ALARM CLOCK BAKELIKE GREEN)  ...   3.791383
1   (ALARM CLOCK BAKELIKE PINK)  ...   3.283859
2    (ALARM CLOCK BAKELIKE RED)  ...   5.568878
3  (ALARM CLOCK BAKELIKE GREEN)  ...   4.916181
4    (ALARM CLOCK BAKELIKE RED)  ...   4.153061

[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
6        (PLASTERS IN TIN SPACEBOY)  ...   2.011670
10    (RED RETROSPOT CHARLOTTE BAG)  ...   5.587746

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

F. Create a Network Routing Diagram

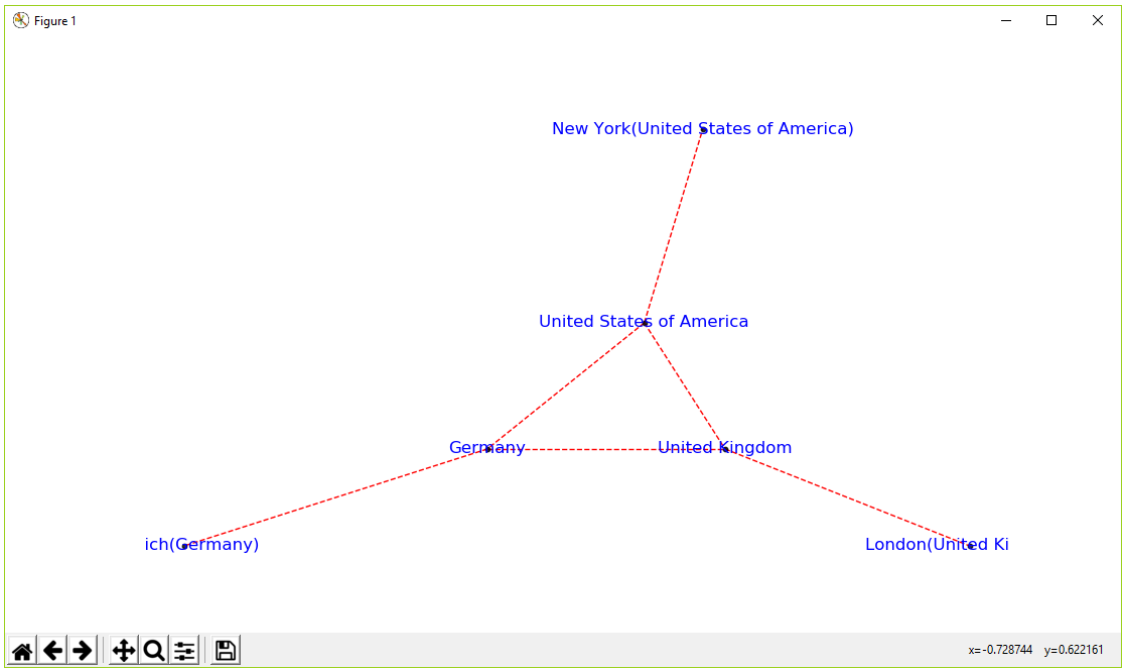
Code:

```
#################################################################
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-Company.csv'
#################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Network-Routing-
Company.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Network-Routing-
Company.png'
Company='01-Vermeulen'
#################################################################
#################################################################
### Import Country Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('#################################')
#################################################################
print(CompanyData.head())
print(CompanyData.shape)
#################################################################
G=nx.Graph()
for i in range(CompanyData.shape[0]):
    for j in range(CompanyData.shape[0]):
        Node0=CompanyData['Company_Country_Name'][i]
        Node1=CompanyData['Company_Country_Name'][j]
        if Node0 != Node1:
            G.add_edge(Node0,Node1)
```

```
for i in range(CompanyData.shape[0]):
   Node0=CompanyData['Company_Country_Name'][i]
   Node1=CompanyData['Company_Place_Name'][i] + '('+
CompanyData['Company_Country_Name'][i] + ')'
   if Node0 != Node1:
      G.add_edge(Node0,Node1)

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=(15, 15))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
#####################################################################
print('##################################')
print('### Done!! ####################')
print('##################################')
#####################################################################
```

Output:

```
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routin
g-Company.csv
################################
################################
  Company_Country_Code  ...      Company_Country_Name
0                   US  ...  United States of America
1                   US  ...  United States of America
2                   US  ...  United States of America
3                   US  ...  United States of America
4                   US  ...  United States of America

[5 rows x 5 columns]
(150, 5)
Nodes: 6
Edges: 6
################################
Storing : C:/VKHCG/01-Vermeulen/05-Organise/01-EDS/02-Python/Organise-Network-Ro
uting-Company.gml
################################
################################
Storing Graph Image: C:/VKHCG/01-Vermeulen/05-Organise/01-EDS/02-Python/Organise
-Network-Routing-Company.png
################################
################################
### Done!! ####################
################################
>>>
```



G. Picking Content for Billboards

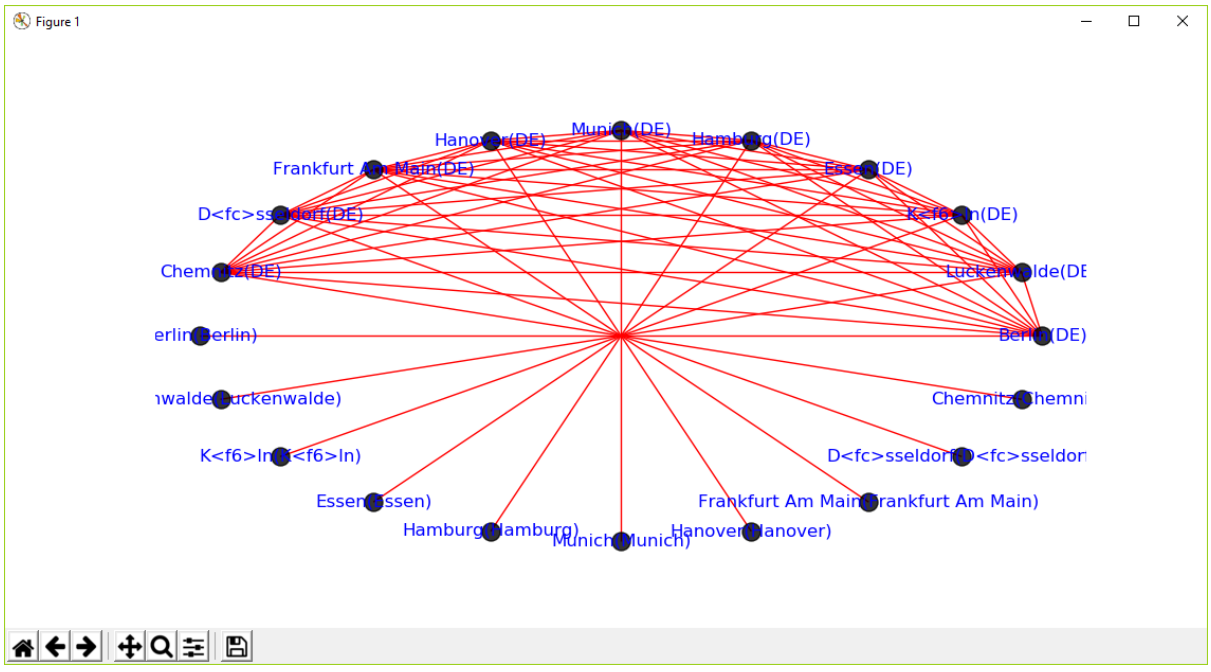
Code:

```
#####################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
#####################################################################
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-DE-Billboard-Visitor.csv'
#####################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Billboards.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Billboards.png'
Company='02-Krennwallner'
#####################################################################
#####################################################################
### Import Company Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
BillboardDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
print('##################################')
#####################################################################
print(BillboardDataRaw.head())
print(BillboardDataRaw.shape)
BillboardData=BillboardDataRaw
sSample=list(np.random.choice(BillboardData.shape[0],20))
#####################################################################
G=nx.Graph()
for i in sSample:
    for j in sSample:
        Node0=BillboardData['BillboardPlaceName'][i] + '('+
BillboardData['BillboardCountry'][i] + ')'
```

```
        Node1=BillboardData['BillboardPlaceName'][j] + '('+
BillboardData['BillboardCountry'][i] + ')'
        if Node0 != Node1:
            G.add_edge(Node0,Node1)

for i in sSample:
    Node0=BillboardData['BillboardPlaceName'][i] + '('+
BillboardData['VisitorPlaceName'][i] + ')'
    Node1=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorCountry'][i] +
')'
    if Node0 != Node1:
        G.add_edge(Node0,Node1)

print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
#####################################################################
sFileName=Base + '/02-Krennwallner/' + sOutputFileName1
print('################################')
print('Storing :',sFileName)
print('################################')
nx.write_gml(G, sFileName)
#####################################################################
sFileName=Base + '/02-Krennwallner/' + sOutputFileName2
print('################################')
print('Storing Graph Image:',sFileName)
print('################################')
plt.figure(figsize=(15, 15))
pos=nx.circular_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=150, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='solid')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
#####################################################################
print('################################')
print('### Done!! ####################')
print('################################')
#####################################################################
```

Output:

```
 RESTART: D:\D-Drive\SIES\Msc IT\Msc IT (Part-1)\Practical Data Science\VKHCG\02-Krennwallner\05-Or
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/02-Krennwallner/02-Assess/01-EDS/02-Python/Assess-DE-Billboard-Visitor.csv
################################
################################
  BillboardCountry BillboardPlaceName  ...  VisitorLongitude  VisitorYearRate
0               DE               Lake  ...           8.5667       26823960.0
1               DE               Horb  ...           8.6833       26823960.0
2               DE               Horb  ...           8.6833       53753112.0
3               DE               Horb  ...           8.6833      107611416.0
4               DE               Horb  ...           8.6833       13359384.0

[5 rows x 9 columns]
(181235, 9)
Nodes: 20
Edges: 55
################################
Storing : C:/VKHCG/02-Krennwallner/05-Organise/01-EDS/02-Python/Organise-Billboards.gml
################################
################################
Storing Graph Image: C:/VKHCG/02-Krennwallner/05-Organise/01-EDS/02-Python/Organise-Billboards.png
################################
################################
### Done!! ######################
################################
>>>
```



H. Create a Delivery Route

Code:

```
import sys
import os
import pandas as pd
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sInputFileName='02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.txt'
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Routes.csv'
Company='03-Hillman'
sFileName=Base + '/' + Company + '/' + sInputFileName
print('Loading :',sFileName)
print('##################################')
RouteDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, sep='|',
encoding="latin-1")
print('##################################')
RouteStart=RouteDataRaw[RouteDataRaw['StartAt']=='WH-KA13']
RouteDistance=RouteStart[RouteStart['Cost']=='DistanceMiles']
RouteDistance=RouteDistance.sort_values(by=['Measure'], ascending=False)
RouteMax=RouteStart["Measure"].max()
RouteMaxCost=round((((RouteMax/1000)*1.5*2)),2)
print('##################################')
print('Maximum (£) per day:')
print(RouteMaxCost)
print('##################################')
RouteMean=RouteStart["Measure"].mean()
RouteMeanMonth=round((((RouteMean/1000)*2*30)),6)
print('##################################')
print('Mean per Month (Miles):')
print(RouteMeanMonth)
print('##################################')
```

Output:

```
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/03-Hillman/02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.txt
################################
################################
################################
Maximum (£) per day:
21.82
################################
################################
Mean per Month (Miles):
21.56191
################################
>>>
```

I. Simple Forex Trading Planner

Code:

```
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
import re
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
sInputFileName='03-Process/01-EDS/02-Python/Process_ExchangeRates.csv'
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Forex.csv'
Company='04-Clark'
sDatabaseName=Base + '/' + Company + '/05-Organise/SQLite/clark.db'
conn = sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
#####################################################################
#####################################################################
### Import Forex Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('#################################')
#####################################################################
ForexDataRaw.index.names = ['RowID']
sTable='Forex_All'
print('Storing :',sDatabaseName,' Table:',sTable)
ForexDataRaw.to_sql(sTable, conn, if_exists="replace")
#####################################################################
sSQL="SELECT 1 as Bag\
      , CAST(min(Date) AS VARCHAR(10)) as Date \
      ,CAST(1000000.0000000 as NUMERIC(12,4)) as Money \
      ,'USD' as Currency \
      FROM Forex_All \
      ;"
sSQL=re.sub("\s\s+", " ", sSQL)
nMoney=pd.read_sql_query(sSQL, conn)
nMoney.index.names = ['RowID']
sTable='MoneyData'
print('Storing :',sDatabaseName,' Table:',sTable)
```

```
nMoney.to_sql(sTable, conn, if_exists="replace")
sTable='TransactionData'
print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
ForexDay=pd.read_sql_query("SELECT Date FROM Forex_All GROUP BY Date;", conn)
t=0
for i in range(ForexDay.shape[0]):
    sDay=ForexDay['Date'][i]
    sSQL='\
    SELECT M.Bag as Bag, \
          F.Date as Date, \
          round(M.Money * F.Rate,6) AS Money, \
          F.CodeIn AS PCurrency, \
          F.CodeOut AS Currency \
    FROM MoneyData AS M \
    JOIN \
    ( \
       SELECT \
       CodeIn, CodeOut, Date, Rate \
       FROM \
       Forex_All \
       WHERE\
       CodeIn = "USD" AND CodeOut = "GBP" \
       UNION \
       SELECT \
       CodeOut AS CodeIn, CodeIn AS CodeOut,  Date, (1/Rate) AS Rate \
       FROM \
       Forex_All \
       WHERE\
       CodeIn = "USD" AND CodeOut = "GBP" \
    ) AS F \
    ON \
    M.Currency=F.CodeIn \
    AND \
    F.Date ='" + sDay + "';'
    sSQL=re.sub("\s\s+", " ", sSQL)
    ForexDayRate=pd.read_sql_query(sSQL, conn)
    for j in range(ForexDayRate.shape[0]):
       sBag=str(ForexDayRate['Bag'][j])
       nMoney=str(round(ForexDayRate['Money'][j],2))
       sCodeIn=ForexDayRate['PCurrency'][j]
       sCodeOut=ForexDayRate['Currency'][j]
    sSQL='UPDATE MoneyData SET Date= '" + sDay + "', '
    sSQL= sSQL + ' Money = ' + nMoney + ', Currency='" + sCodeOut + "'"
    sSQL= sSQL + ' WHERE Bag=' + sBag + ' AND Currency='" + sCodeIn + "';'
    sSQL=re.sub("\s\s+", " ", sSQL)
    cur = conn.cursor()
    cur.execute(sSQL)
    conn.commit()
    t+=1
```

```
    print('Trade :', t, sDay, sCodeOut, nMoney)

    sSQL=' \
    INSERT INTO TransactionData ( \
                            RowID, \
                            Bag, \
                            Date, \
                            Money, \
                            Currency \
                        ) \
    SELECT ' + str(t) + ' AS RowID, \
      Bag, \
      Date, \
      Money, \
      Currency \
    FROM MoneyData \
    ;'

    sSQL=re.sub("\s\s+", " ", sSQL)
    cur = conn.cursor()
    cur.execute(sSQL)
    conn.commit()
sSQL="SELECT RowID, Bag, Date, Money, Currency FROM TransactionData ORDER BY
RowID;"
sSQL=re.sub("\s\s+", " ", sSQL)
TransactionData=pd.read_sql_query(sSQL, conn)
OutputFile=Base + '/' + Company + '/' + sOutputFileName
TransactionData.to_csv(OutputFile, index = False)
```

Output:

Practical No. 9

Aim: Generating Data

Step 1:
Make sure you are using the latest version of **Python**.
Step 2:
Install all the following modules before performing the practical:

Commands For the modules:

```
pip install folium
```

A. Vermeulen PLC

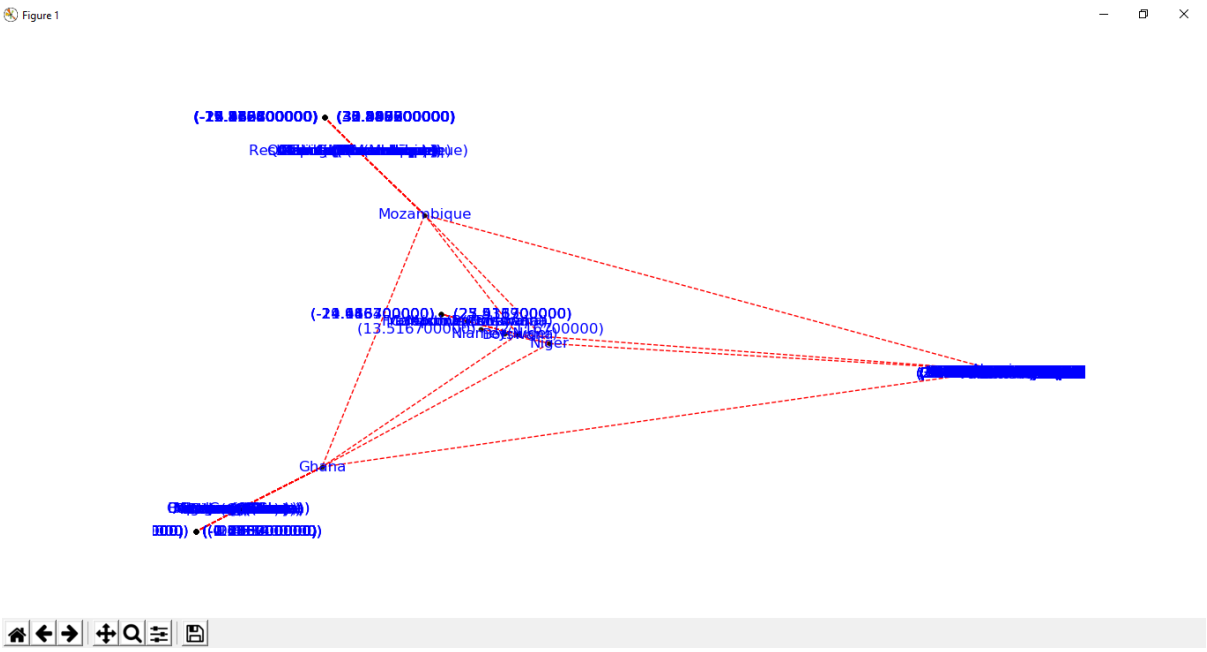
Code:

```
######################################################################
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-Network-Routing-Customer.gml'
sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png'
Company='01-Vermeulen'
######################################################################
######################################################################
### Import Country Data
######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
CustomerDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
CustomerData=CustomerDataRaw.head(100)
print('Loaded Country:',CustomerData.columns.values)
print('##################################')
######################################################################
```

```
print(CustomerData.head())
print(CustomerData.shape)
#####################################################################
G=nx.Graph()
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='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
#####################################################################
print('#################################')
print('### Done!! ####################')
print('#################################')
#####################################################################
```

Output:

```
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing
-Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude
'
 'Customer_Longitude' 'Customer_Country_Name']
################################
  Customer_Country_Code  ... Customer_Country_Name
0                    BW  ...               Botswana
1                    BW  ...               Botswana
2                    BW  ...               Botswana
3                    BW  ...               Botswana
4                    NE  ...                  Niger

[5 rows x 5 columns]
(100, 5)
Nodes: 205
Edges: 210
################################
Storing : C:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Network-Routing
-Customer.gml
################################
################################
Storing Graph Image: C:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Netw
ork-Routing-Customer.png
################################
################################
### Done!! ######################
################################
>>>
```



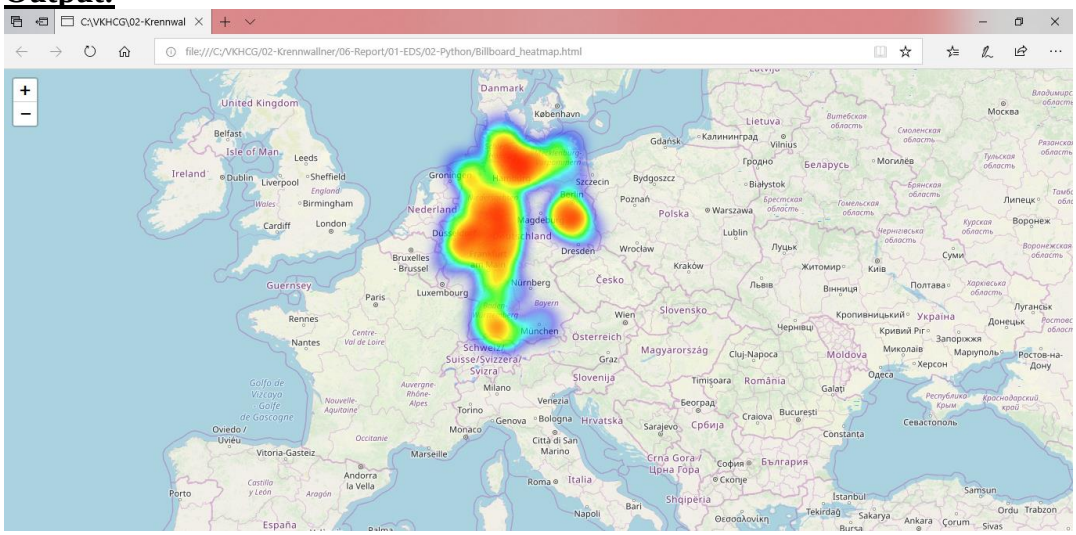
B. Picking Content for Billboards

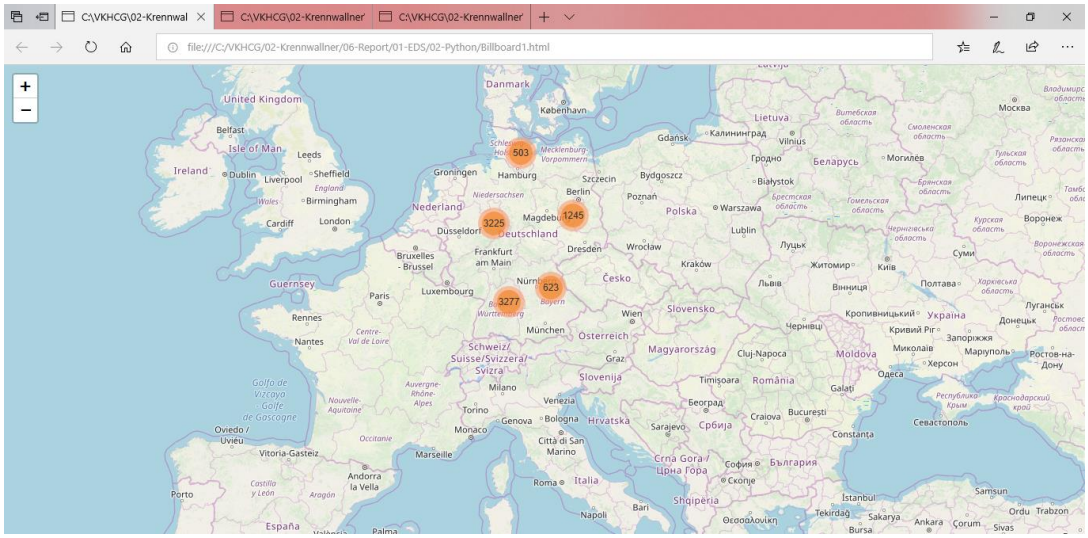
Code:

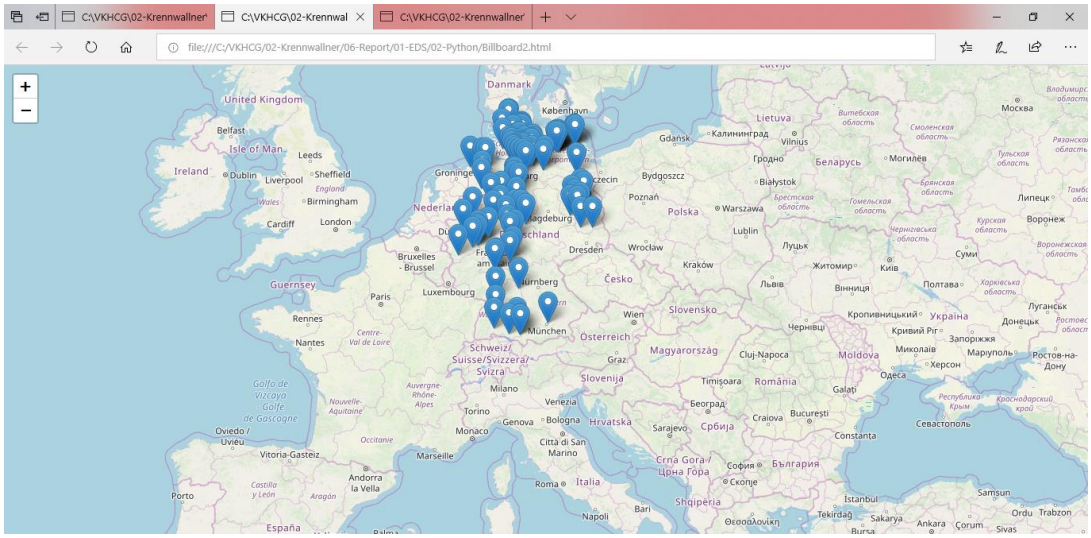
```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
from folium.plugins import FastMarkerCluster, HeatMap
from folium import Marker, Map
import 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/02-
Python/Retrieve_DE_Billboard_Locations.csv'
df = pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
df.fillna(value=0, inplace=True)
print(df.shape)
#####################################################################
t=0
for i in range(df.shape[0]):
   try:
      sLongitude=df["Longitude"][i]
      sLongitude=float(sLongitude)
   except Exception:
      sLongitude=float(0.0)

   try:
      sLatitude=df["Latitude"][i]
      sLatitude=float(sLatitude)
   except Exception:
      sLatitude=float(0.0)

   try:
      sDescription=df["Place_Name"][i] + ' (' + df["Country"][i]+')'
   except Exception:
      sDescription='VKHCG'

   if sLongitude != 0.0 and sLatitude != 0.0:
      DataClusterList=list([sLatitude, sLongitude])
      DataPointList=list([sLatitude, sLongitude, sDescription])
```

```
        t+=1
        if t==1:
            DataCluster=[DataClusterList]
            DataPoint=[DataPointList]
        else:
            DataCluster.append(DataClusterList)
            DataPoint.append(DataPointList)
data=DataCluster
pins=pd.DataFrame(DataPoint)
pins.columns = [ 'Latitude','Longitude','Description']
#####################################################################
stops_map1 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
marker_cluster = FastMarkerCluster(data).add_to(stops_map1)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard1.html'
stops_map1.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
#####################################################################
stops_map2 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
for name, row in pins.iloc[:100].iterrows():
    Marker([row["Latitude"],row["Longitude"]],
popup=row["Description"]).add_to(stops_map2)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html'
stops_map2.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
#####################################################################
stops_heatmap = Map(location=[48.1459806, 11.4985484], zoom_start=5)
stops_heatmap.add_child(HeatMap([[row["Latitude"], row["Longitude"]] for name, row in
pins.iloc[:100].iterrows()]))
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-
Python/Billboard_heatmap.html'
stops_heatmap.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

Output:








```
################################
Working Base : C:/VKHCG  using  win32
################################
(8873, 4)
### Done!! ############################################
>>>
```

C. Hillman Ltd

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 title 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))
```

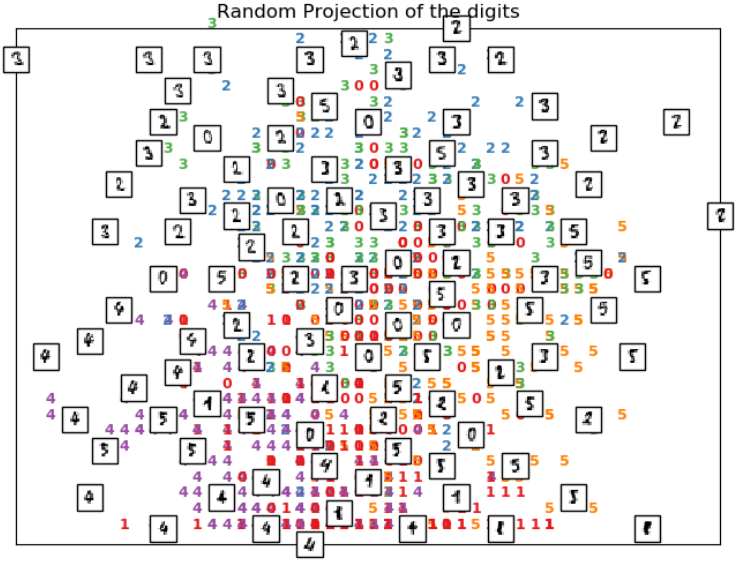
```
print("Computing MDS embedding")
clf = manifold.MDS(n_components=2, n_init=1, max_iter=100)
t0 = time()
X_mds = clf.fit_transform(X)
print("Done. Stress: %f" % clf.stress_)
plot_embedding(X_mds,"MDS embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing Totally Random Trees embedding")
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:

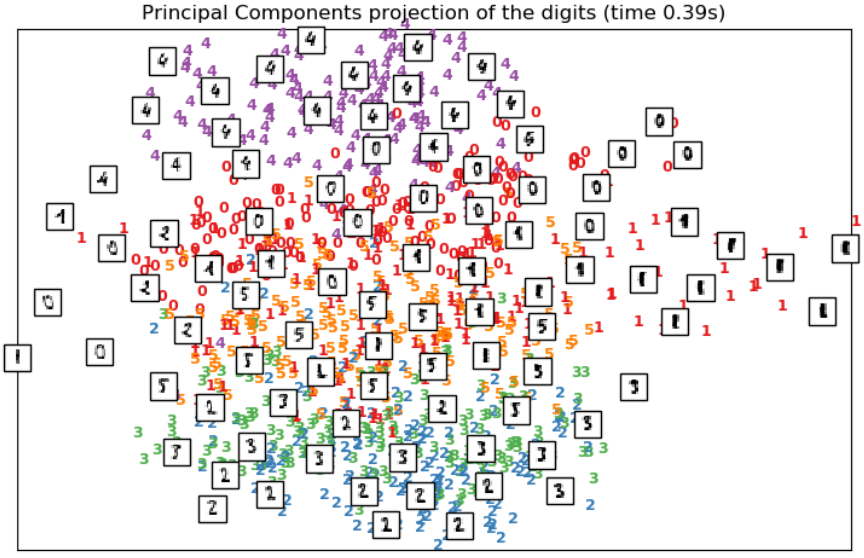
```
Computing random projection
Computing PCA projection
Computing Linear Discriminant Analysis projection
Computing Isomap embedding
Done.
Computing LLE embedding
Done. Reconstruction error: 1.63544e-06
Computing modified LLE embedding
Done. Reconstruction error: 0.360657
Computing Hessian LLE embedding
Done. Reconstruction error: 0.212802
Computing LTSA embedding
Done. Reconstruction error: 0.212804
Computing MDS embedding
Done. Stress: 137741752.343511
Computing Totally Random Trees embedding
Computing Spectral embedding
Computing t-SNE embedding
>>>
```

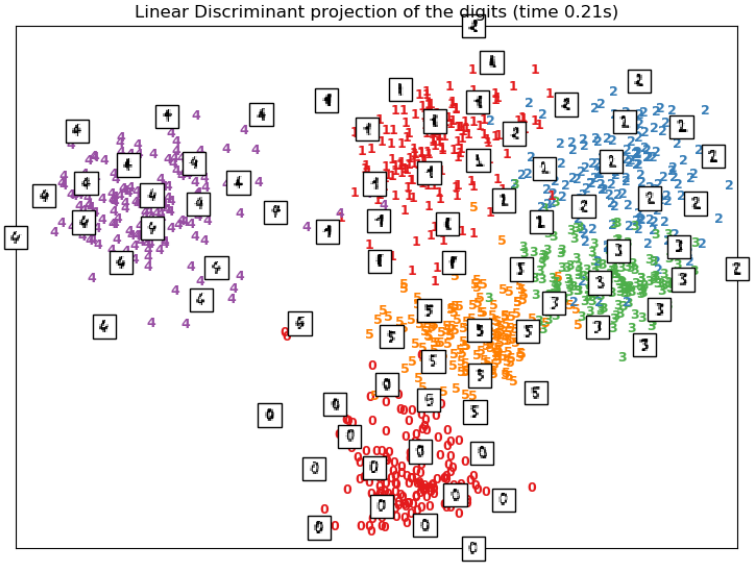
A selection from the 64-dimensional digits dataset

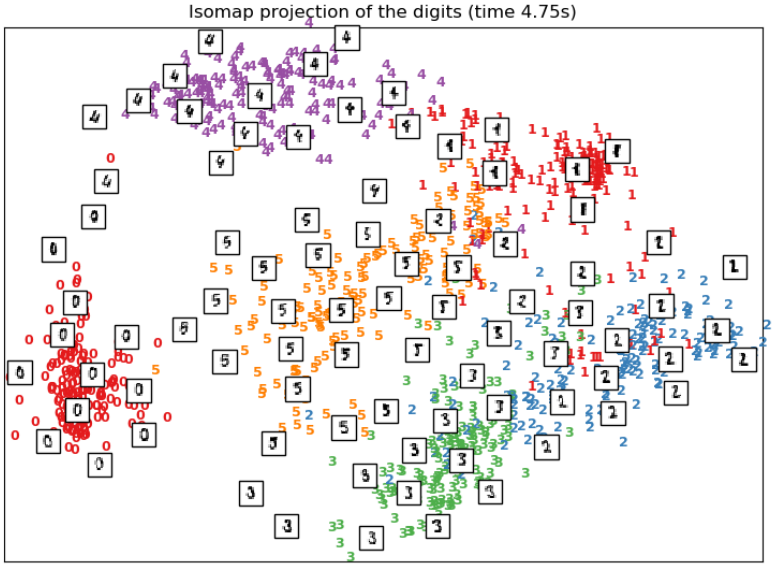
Random Projection of the digits

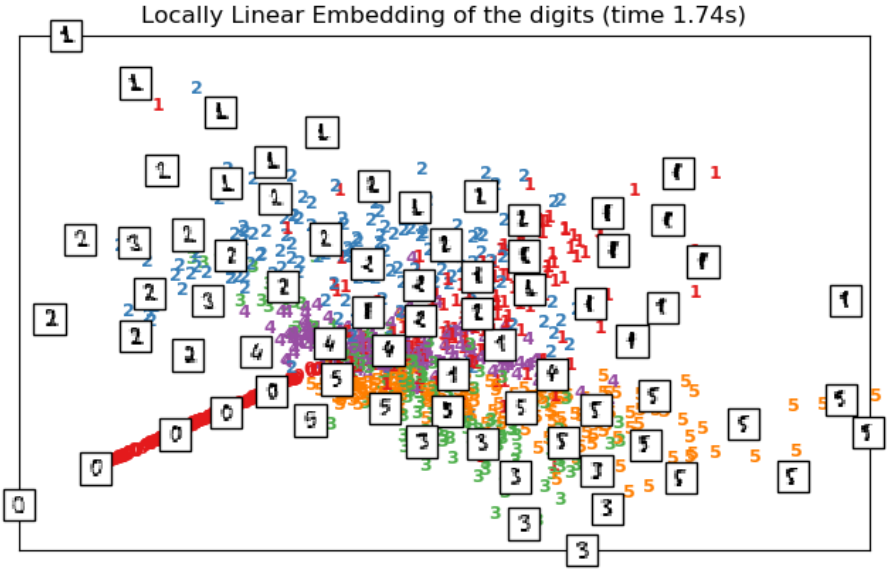


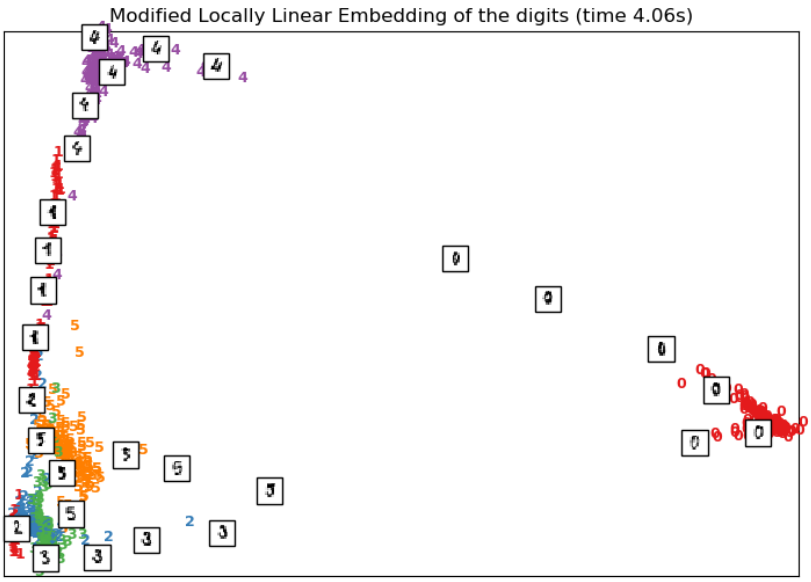
Principal Components projection of the digits (time 0.39s)

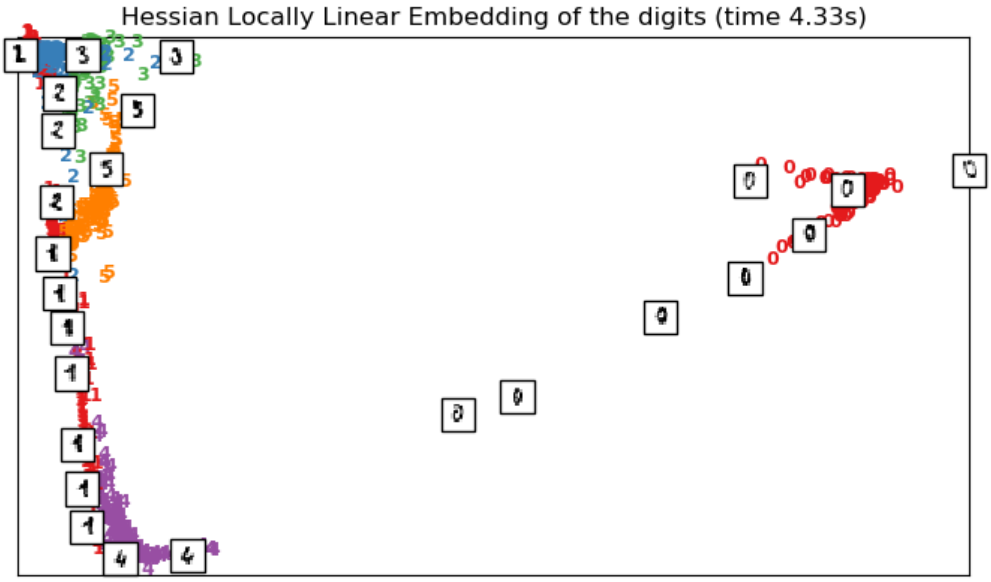


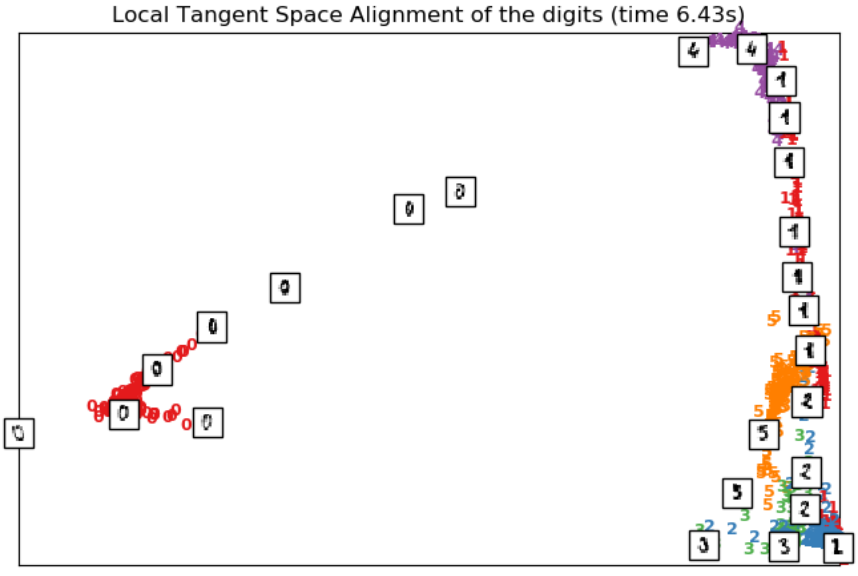

Linear Discriminant projection of the digits (time 0.21s)

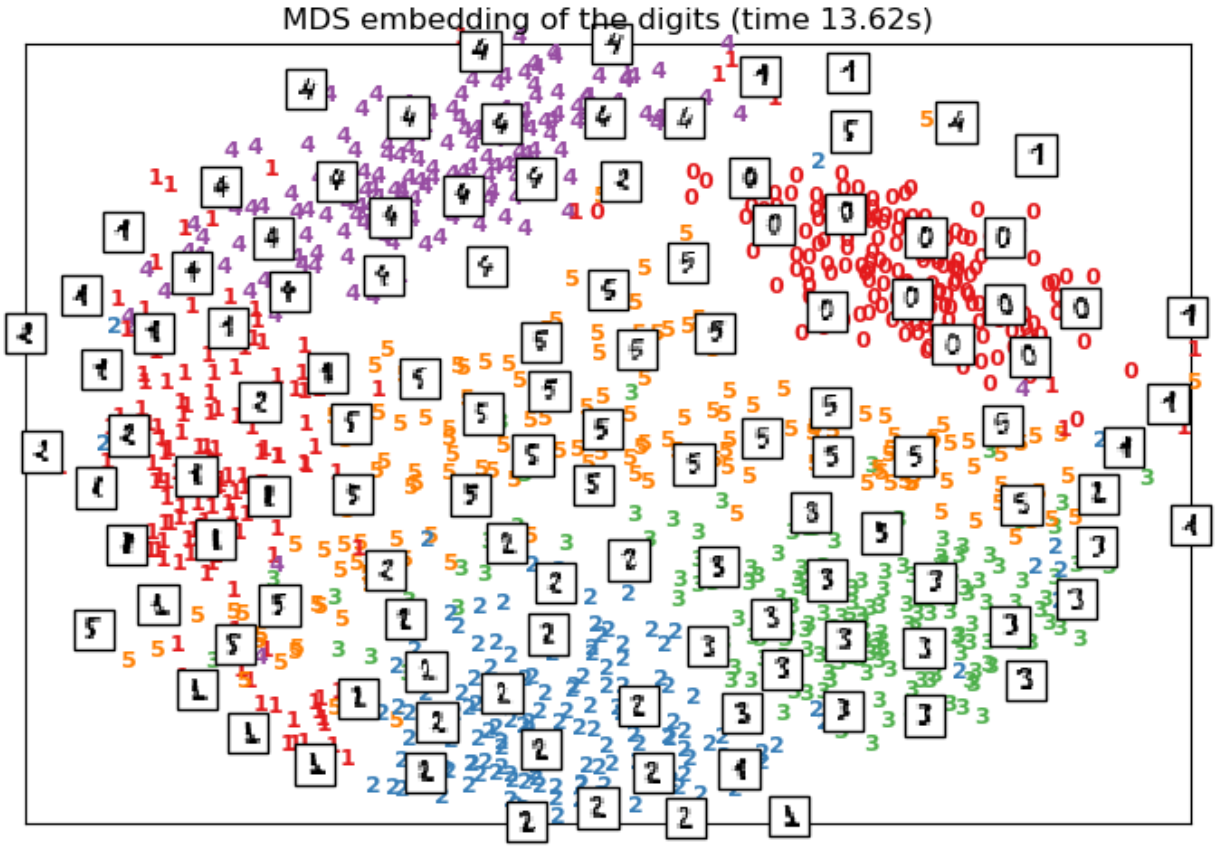

Isomap projection of the digits (time 4.75s)

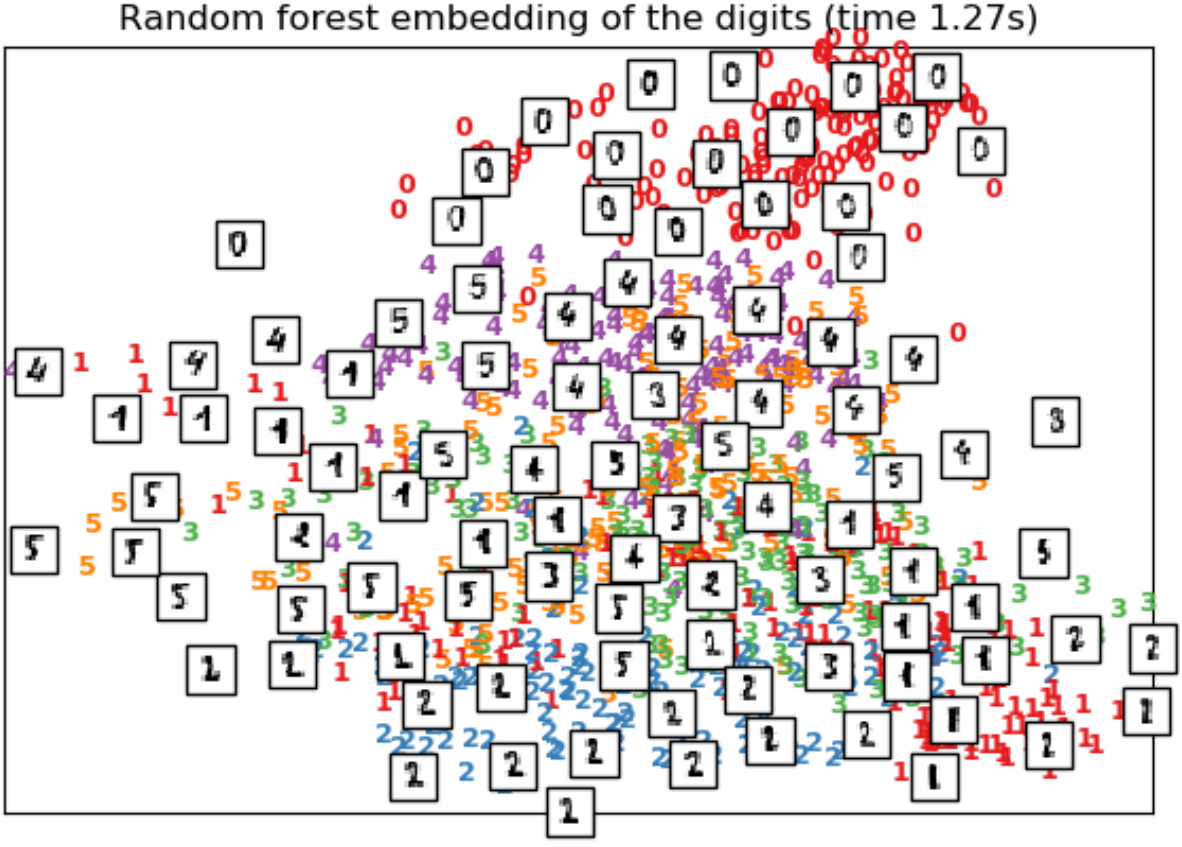

Locally Linear Embedding of the digits (time 1.74s)

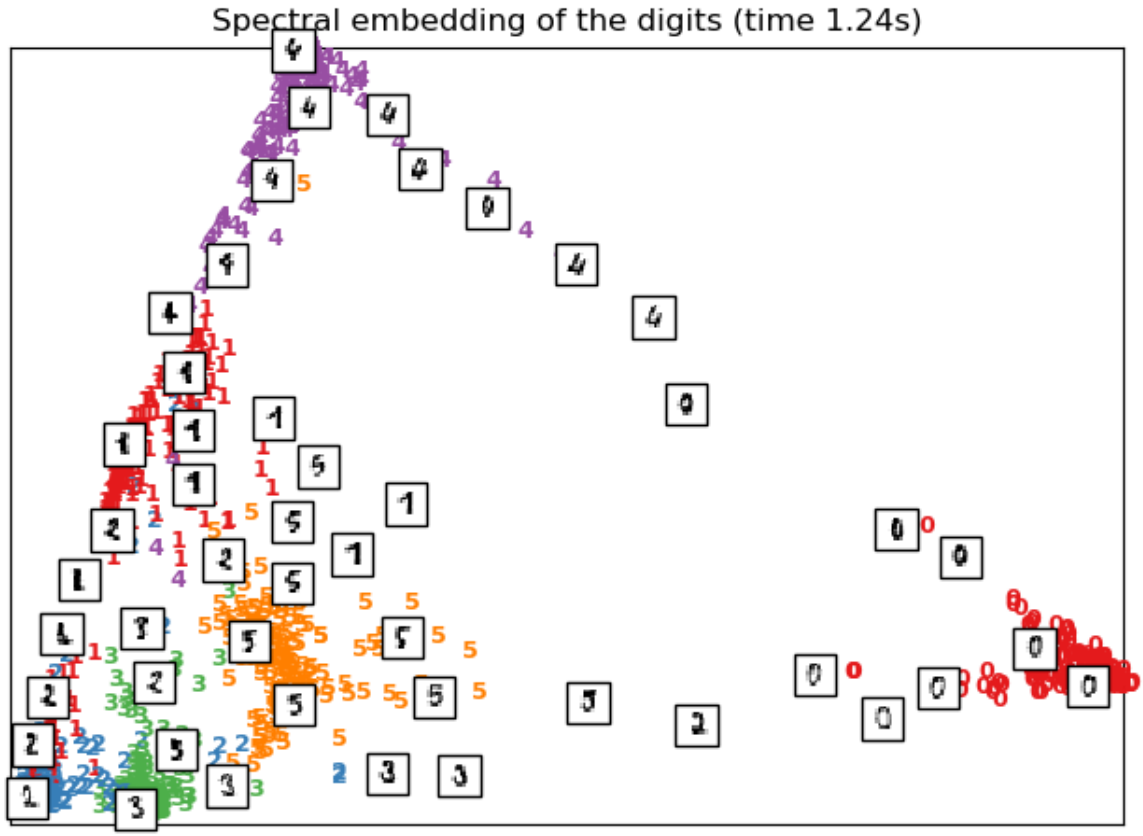

Modified Locally Linear Embedding of the digits (time 4.06s)

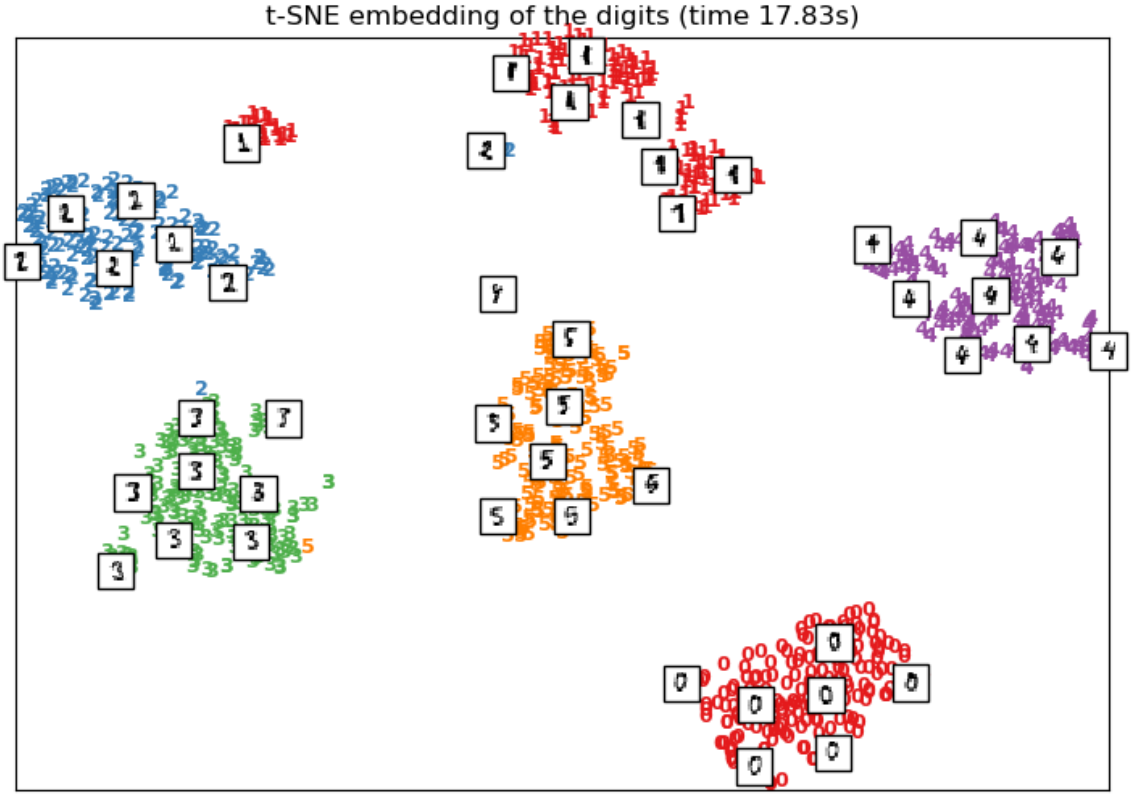

Hessian Locally Linear Embedding of the digits (time 4.33s)


Local Tangent Space Alignment of the digits (time 6.43s)


MDS embedding of the digits (time 13.62s)


Random forest embedding of the digits (time 1.27s)


Spectral embedding of the digits (time 1.24s)


t-SNE embedding of the digits (time 17.83s)

D. Report Balance Sheet

Code:

```
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
import re
from openpyxl import load_workbook
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    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-Balance-Sheet'
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=False, 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)
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)
```

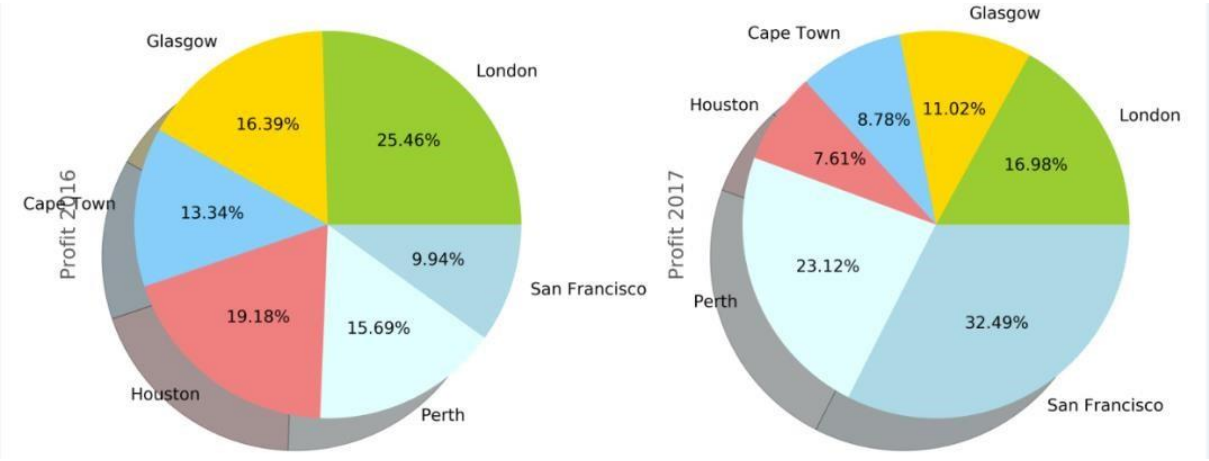
Output:

```
################################
Storing : C:/VKHCG/04-Clark/06-Report/SQLite/clark.db  Table: BalanceSheets
Append Data
(71136, 7)

Warning (from warnings module):
  File "C:\VKHCG\04-Clark\06-Report\Report-Balance-Sheet.py", line 76
    ws=wb.get_sheet_by_name("Balance-Sheet")
DeprecationWarning: Call to deprecated function get_sheet_by_name (Use wb[sheetname]).
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Clark-Afghanistan.xlsx
Set cell D16  to  Cash Total
Set cell D17  to  Accounts_Receivable Total
Set cell H25  to  Other_Long_Term_Liabilities Total
Set cell H30  to  Capital_Stock Total
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Hillman-Afghanistan.xlsx
Set cell D16  to  Cash Total
Set cell D17  to  Accounts_Receivable Total
Set cell H25  to  Other_Long_Term_Liabilities Total
Set cell H30  to  Capital_Stock Total
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Krennwallner-Afghanistan.xlsx
Set cell D16  to  Cash Total
Set cell D17  to  Accounts_Receivable Total
Set cell H25  to  Other_Long_Term_Liabilities Total
Set cell H30  to  Capital_Stock Total
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Vermeulen-Afghanistan.xlsx
Set cell D16  to  Cash Total
Set cell D17  to  Accounts_Receivable Total
Set cell H25  to  Other_Long_Term_Liabilities Total
Set cell H30  to  Capital_Stock Total
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Clark-AlandIslands.xlsx
Set cell D16  to  Cash Total
Set cell D17  to  Accounts_Receivable Total
```

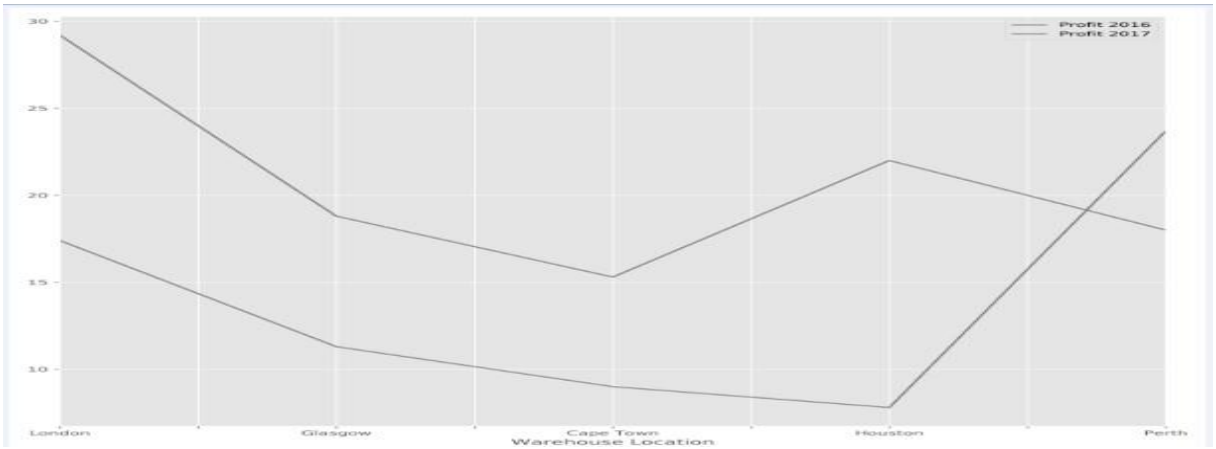
E. Graphics

Pie Graph Double Pie

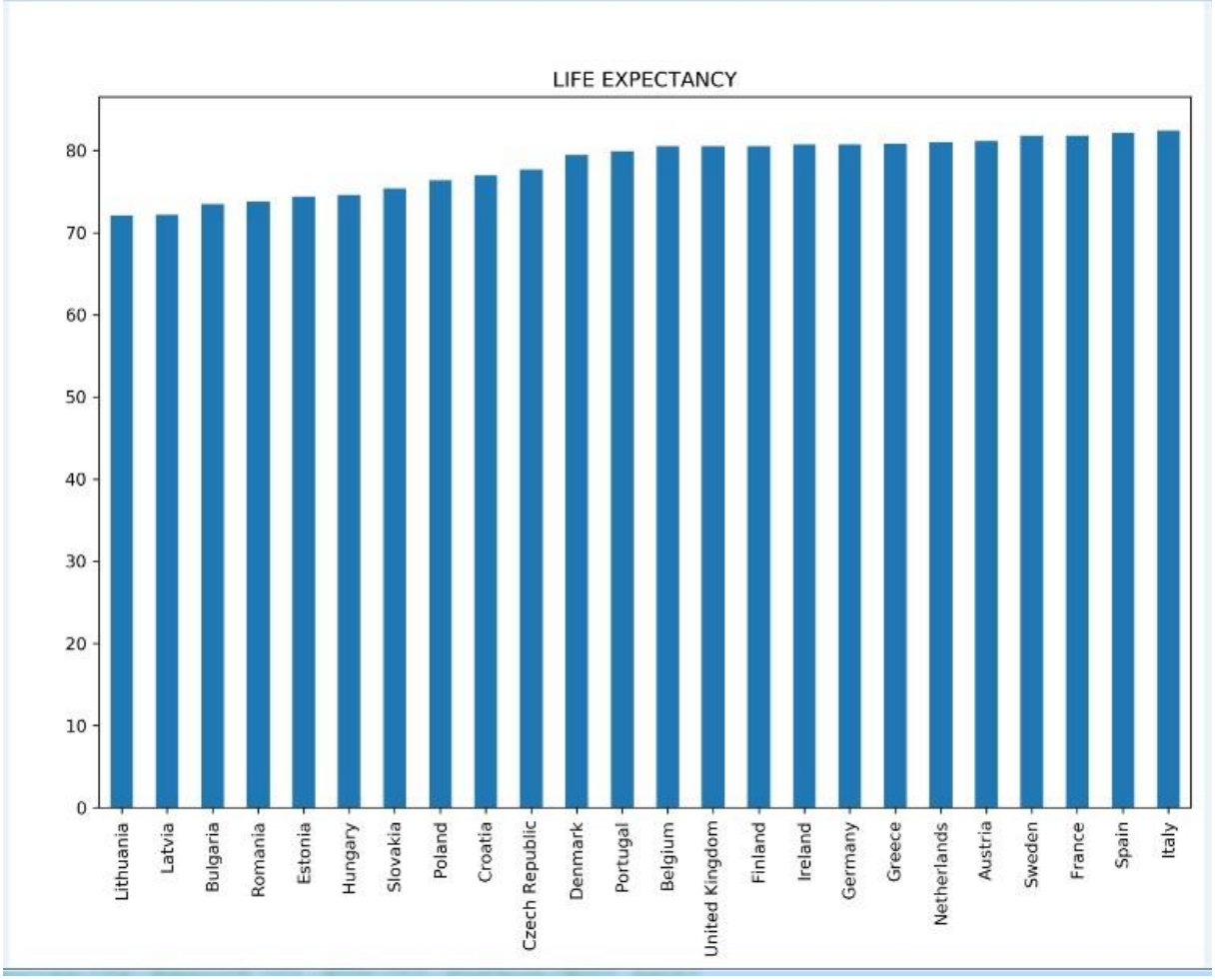
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py



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

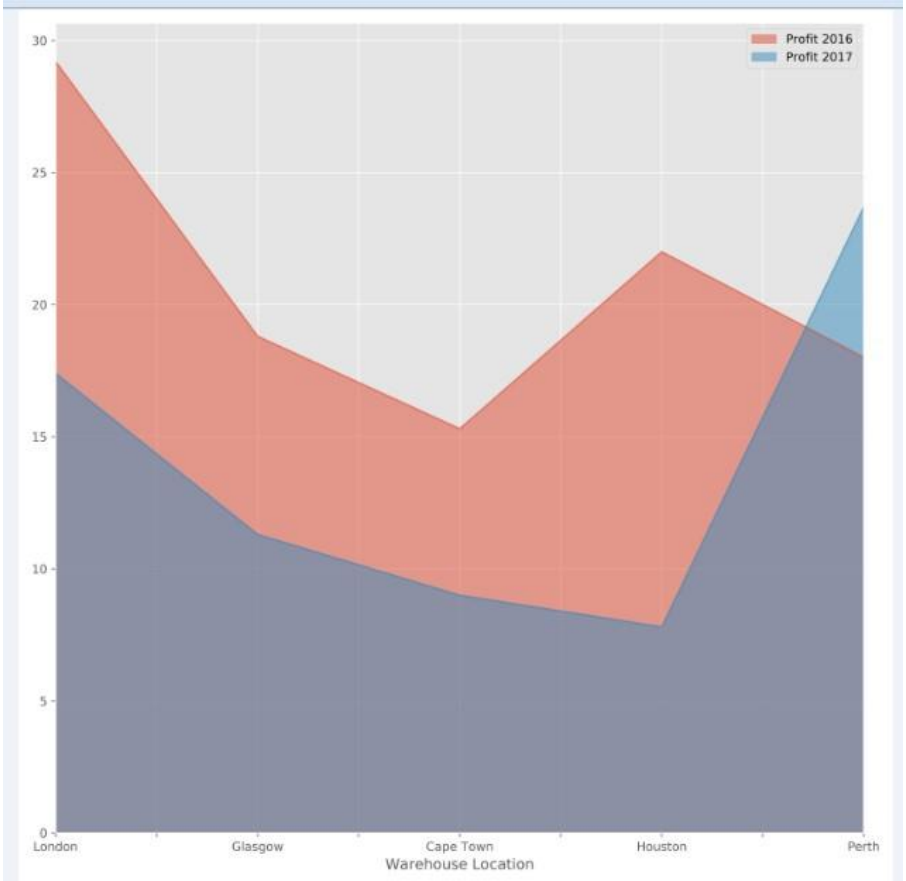


Bar Graph / Horizontal Bar Graph
C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py



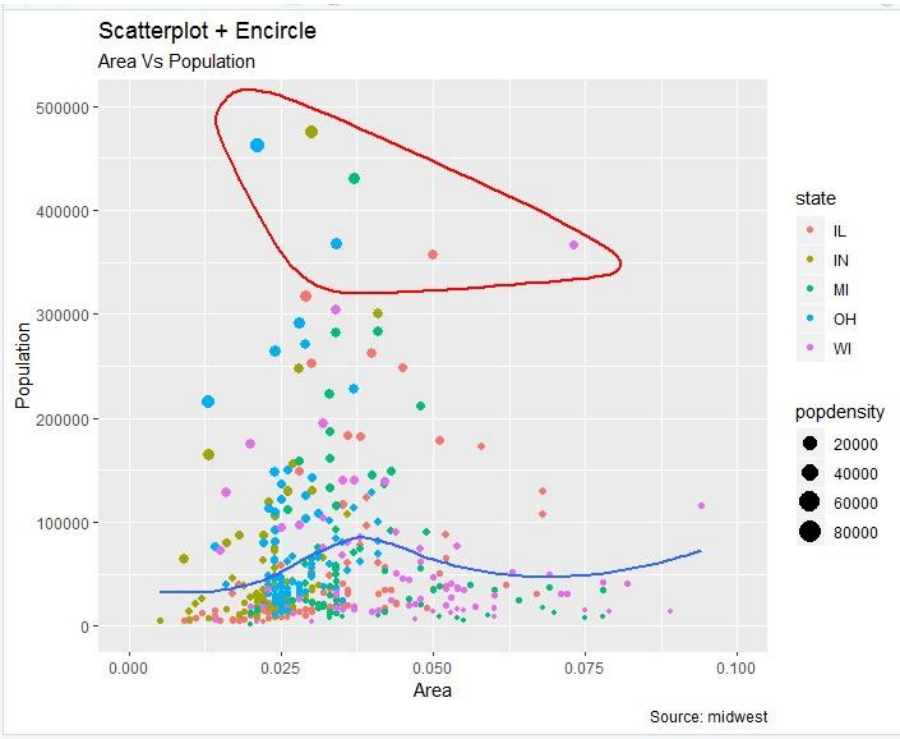
Area Graph

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



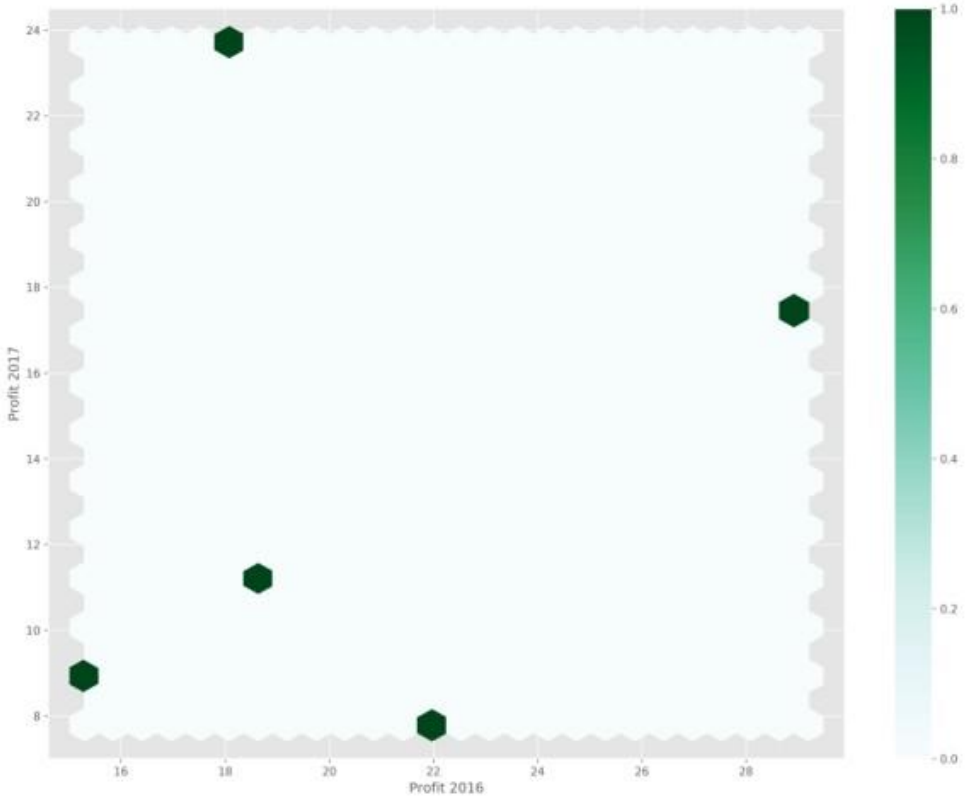
Scatter Graph

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



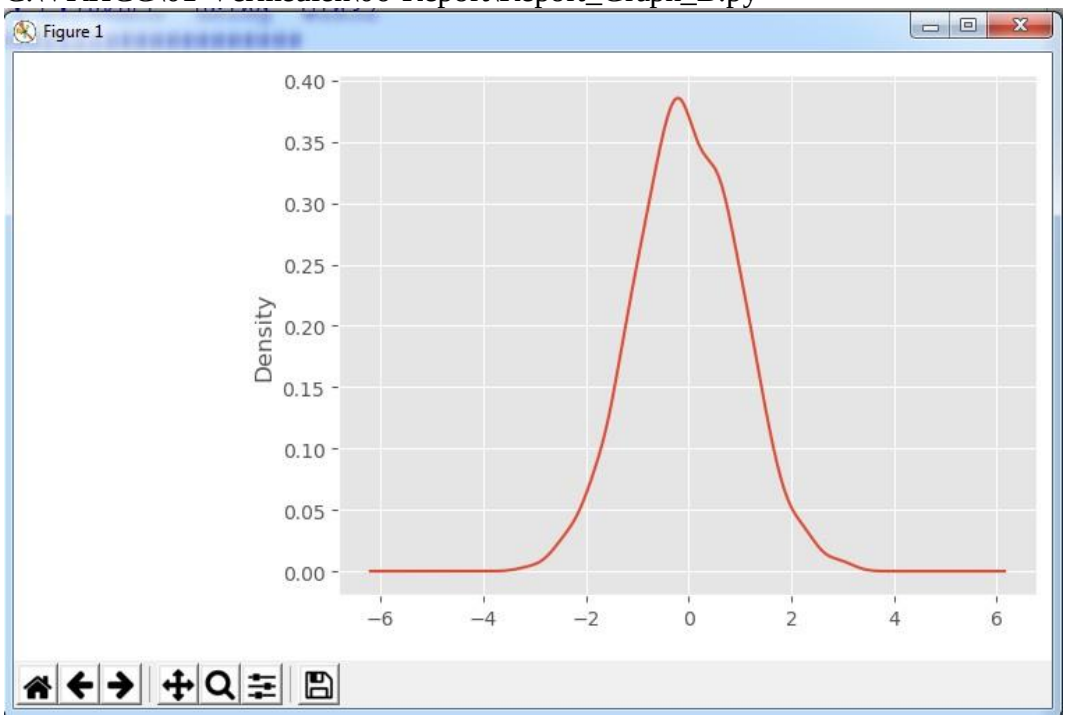
Hexbin

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



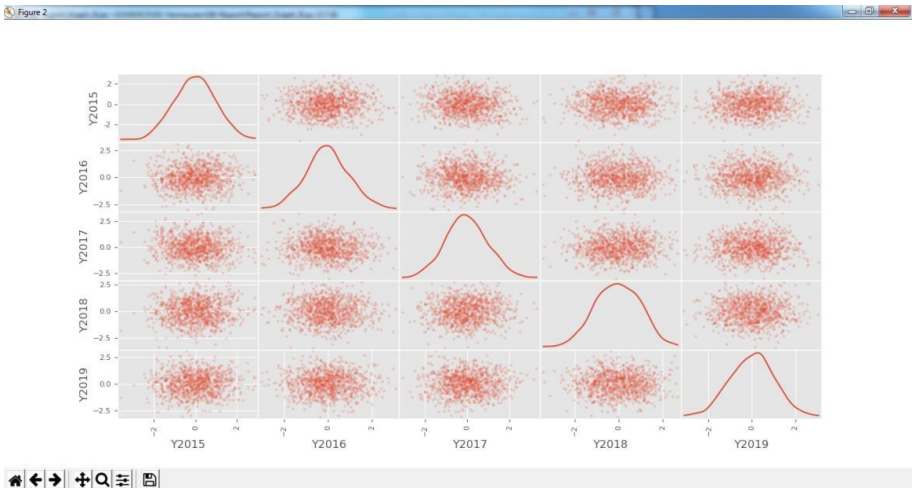
Kernel Density Estimation (KDE) Graph

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



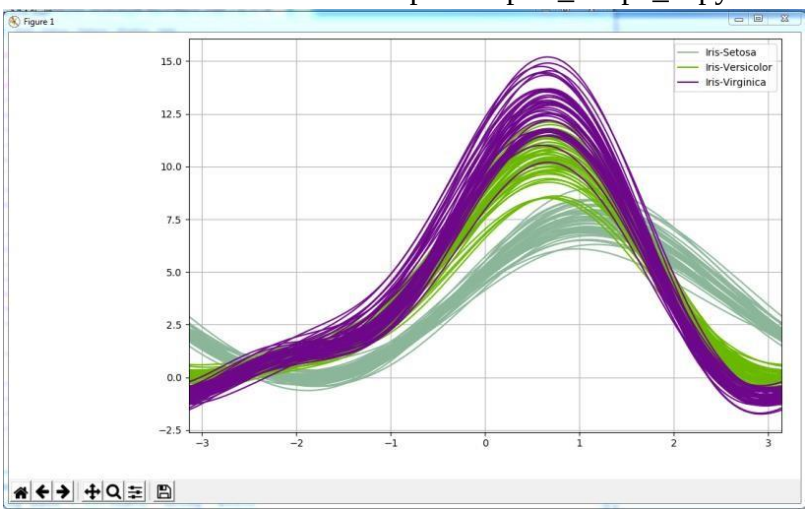
Scatter Matrix Graph

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



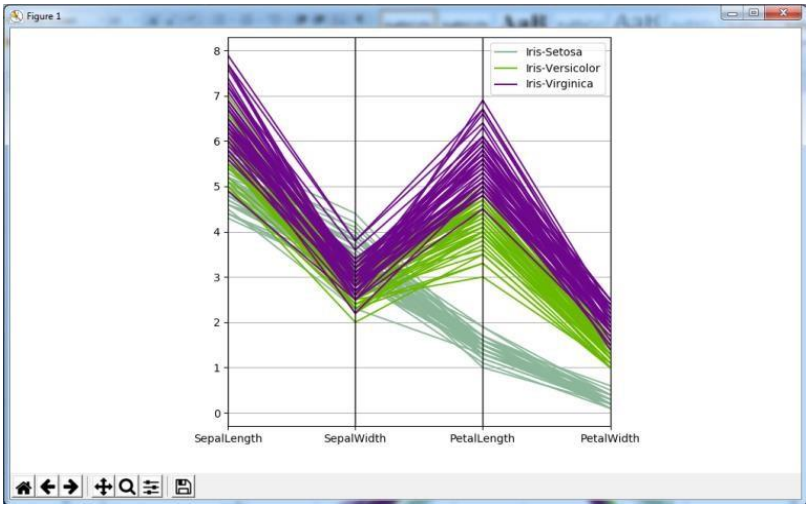
Andrews' Curves

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



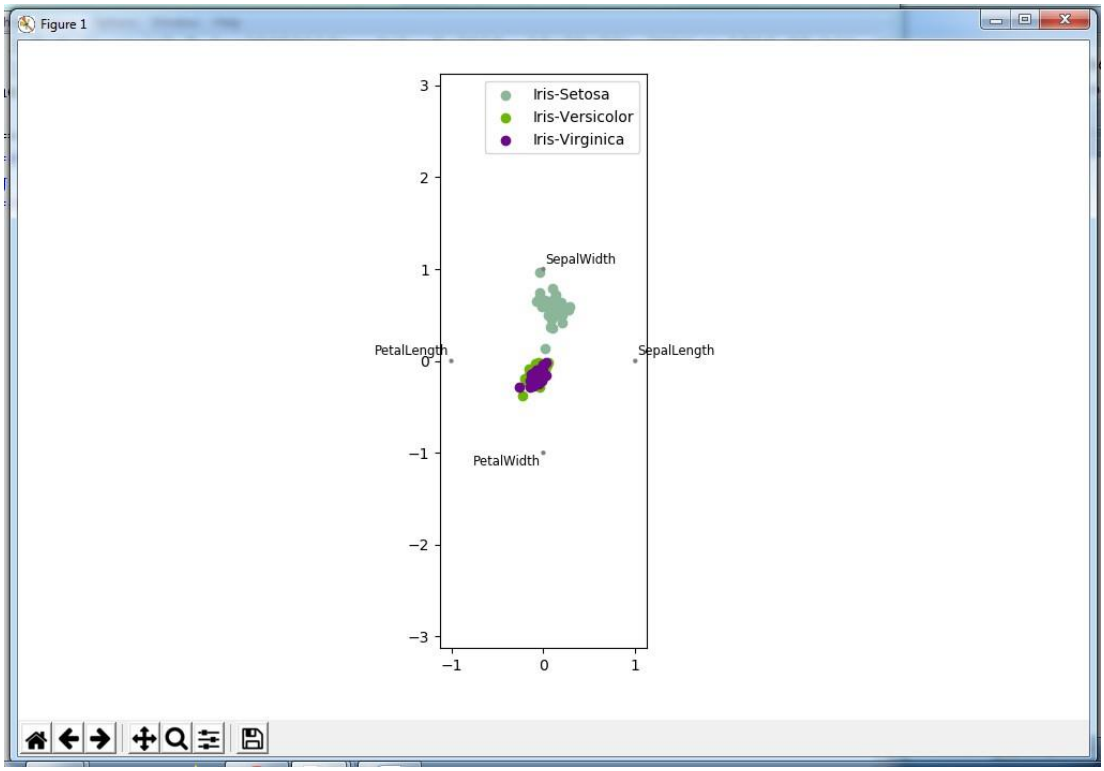
Parallel Coordinates

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



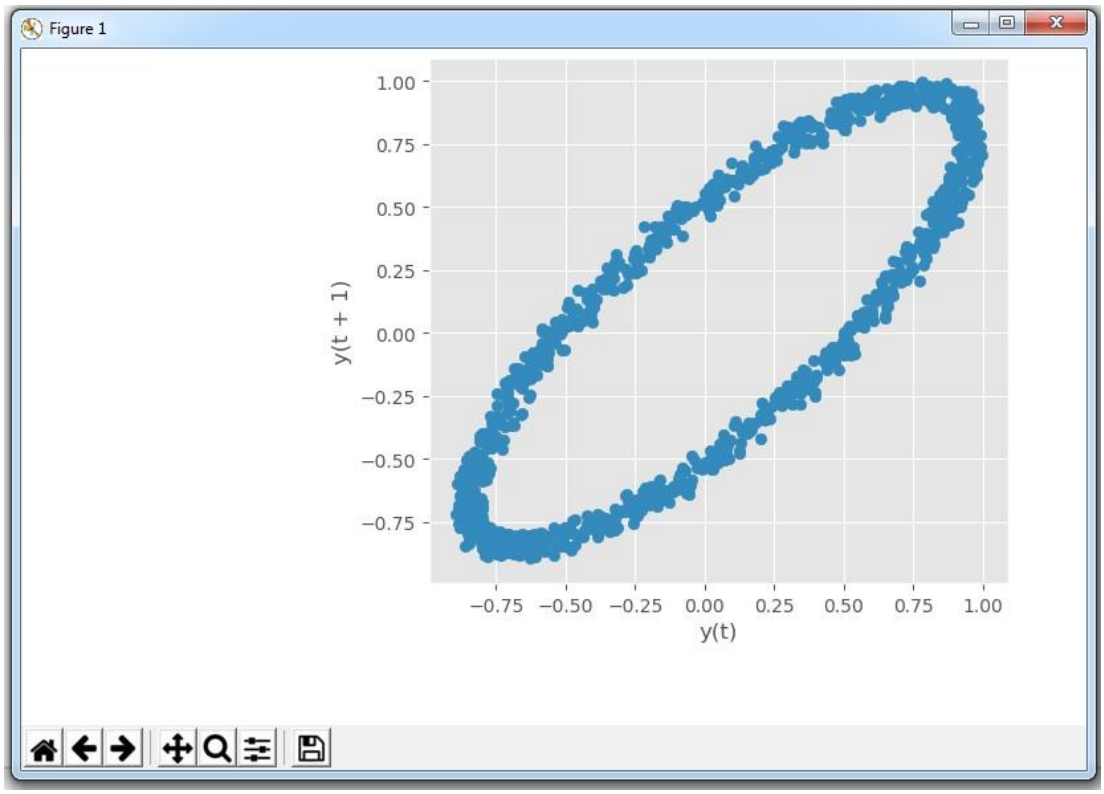
RADVIZ Method

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



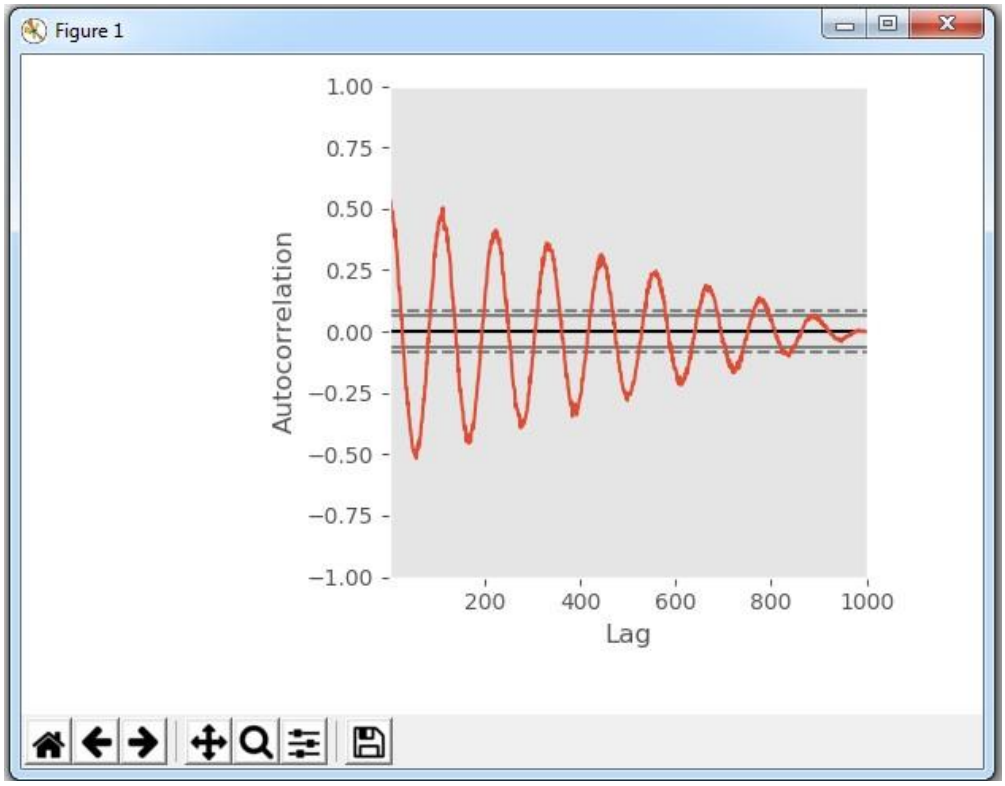
Lag Plot

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



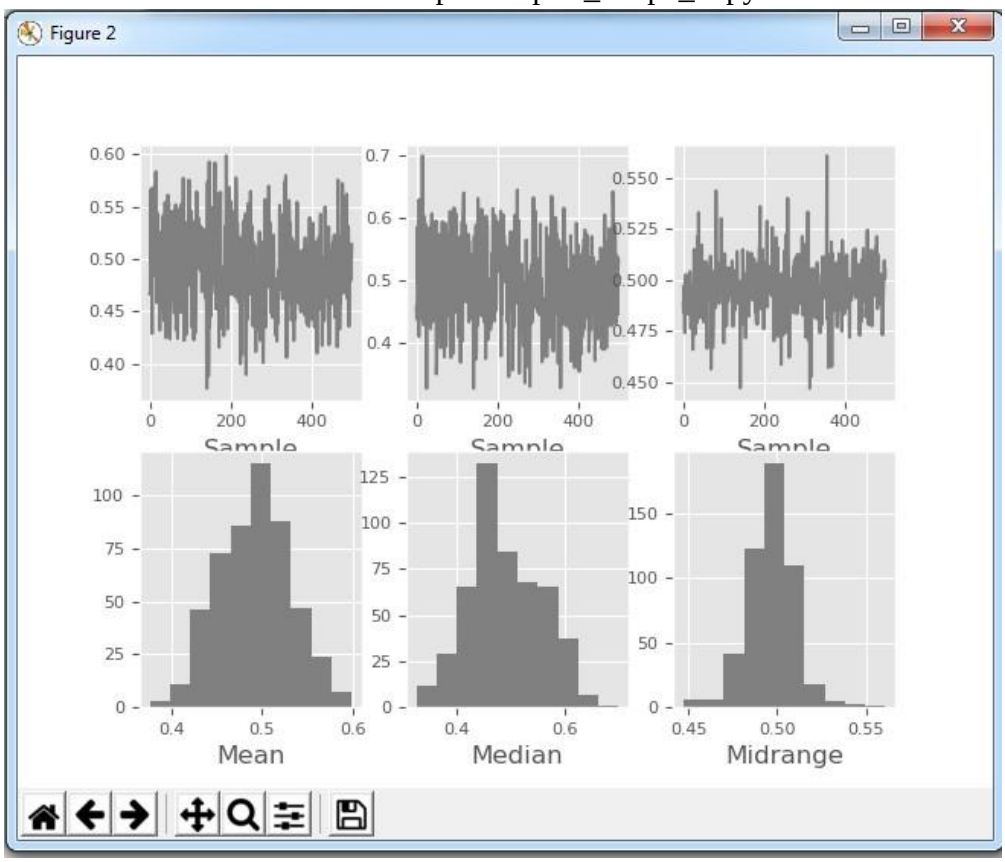
Autocorrelation Plot

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



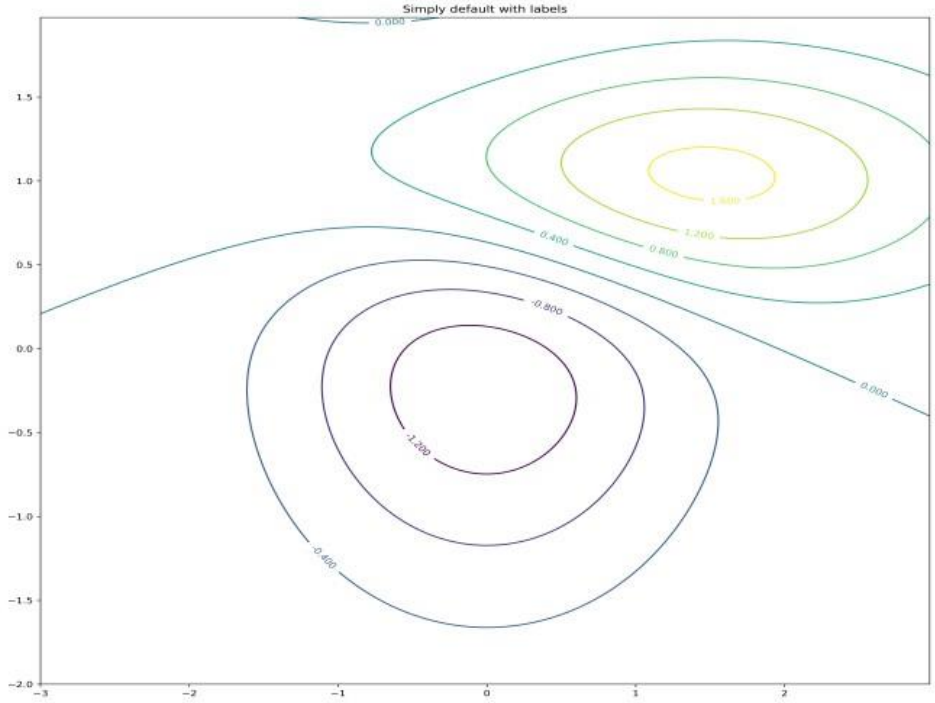
Bootstrap Plot

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



Contour Graphs

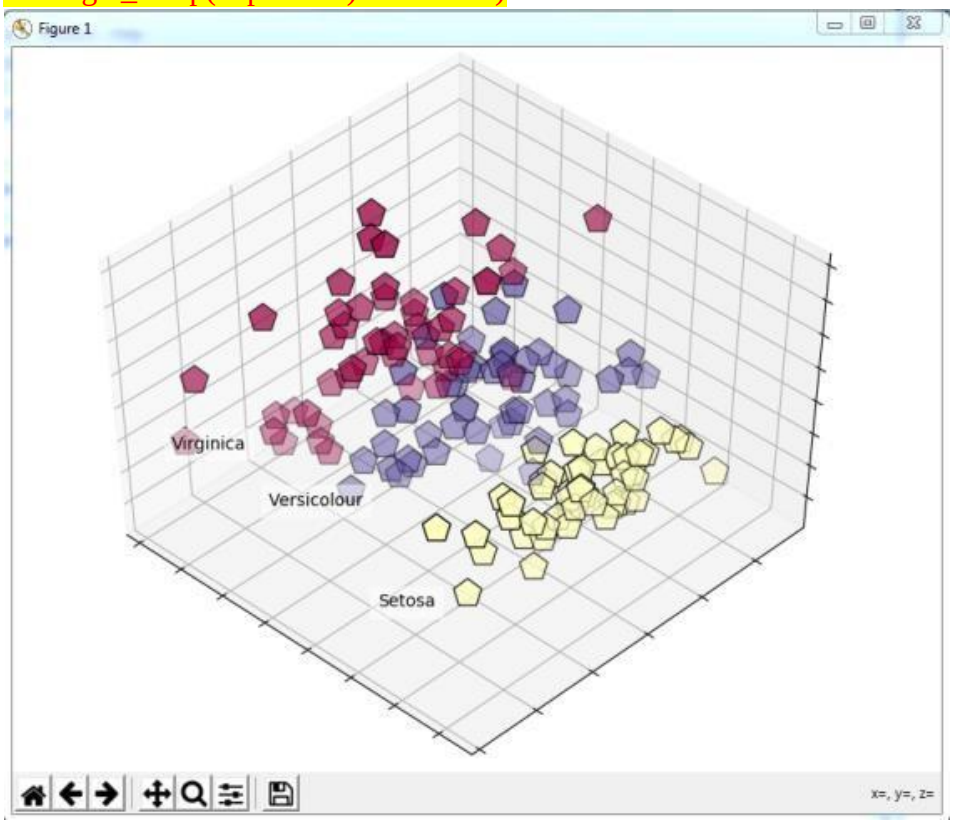
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_G.py



3D Graphs

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

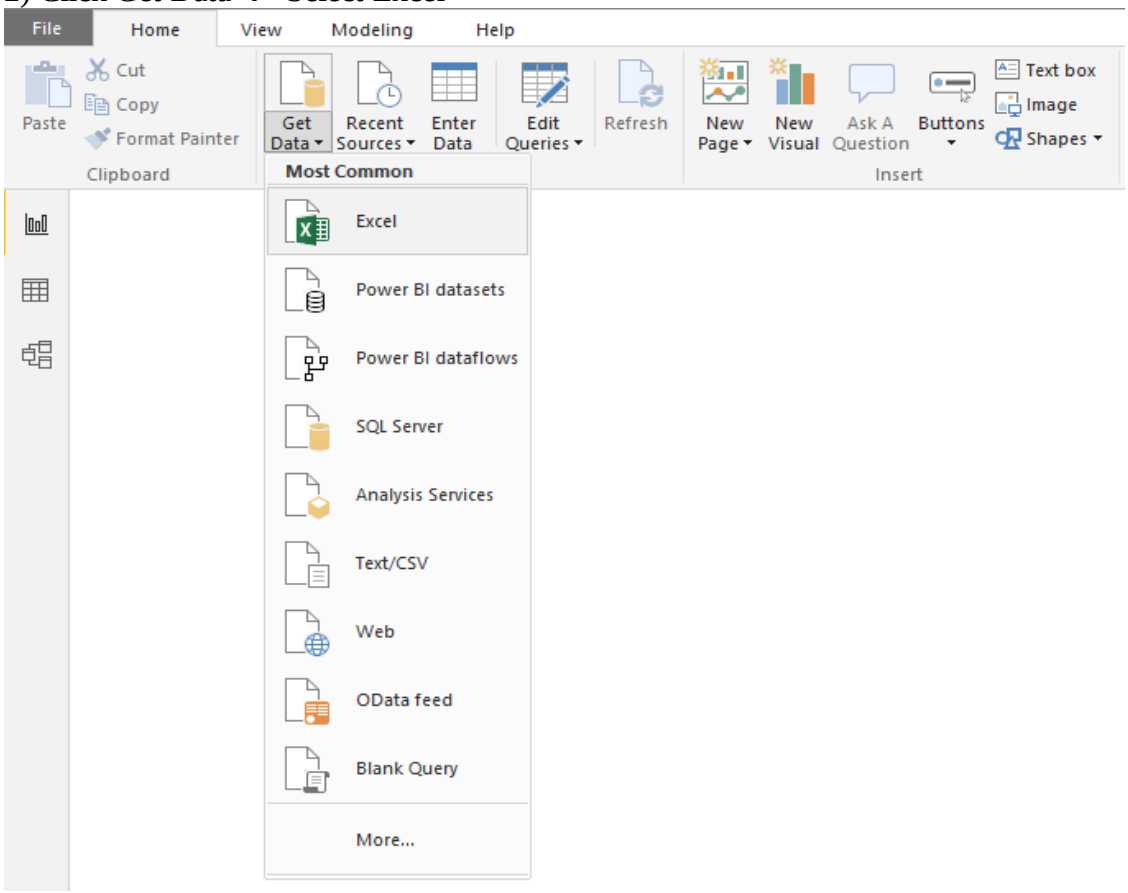
(add →import matplotlib.cm as cm & Replace : plt.cm.spectral →cm.get_cmap("Spectral") at Line 44)



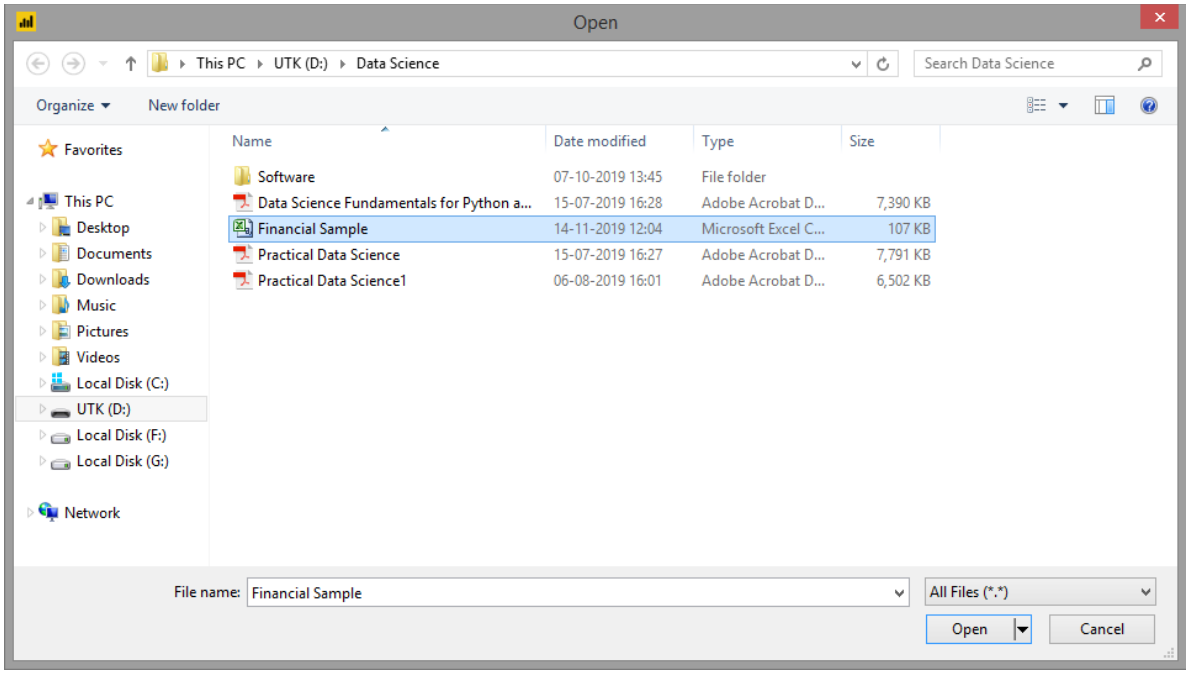
Practical No. 10

Aim: Data Visualization with Power BI

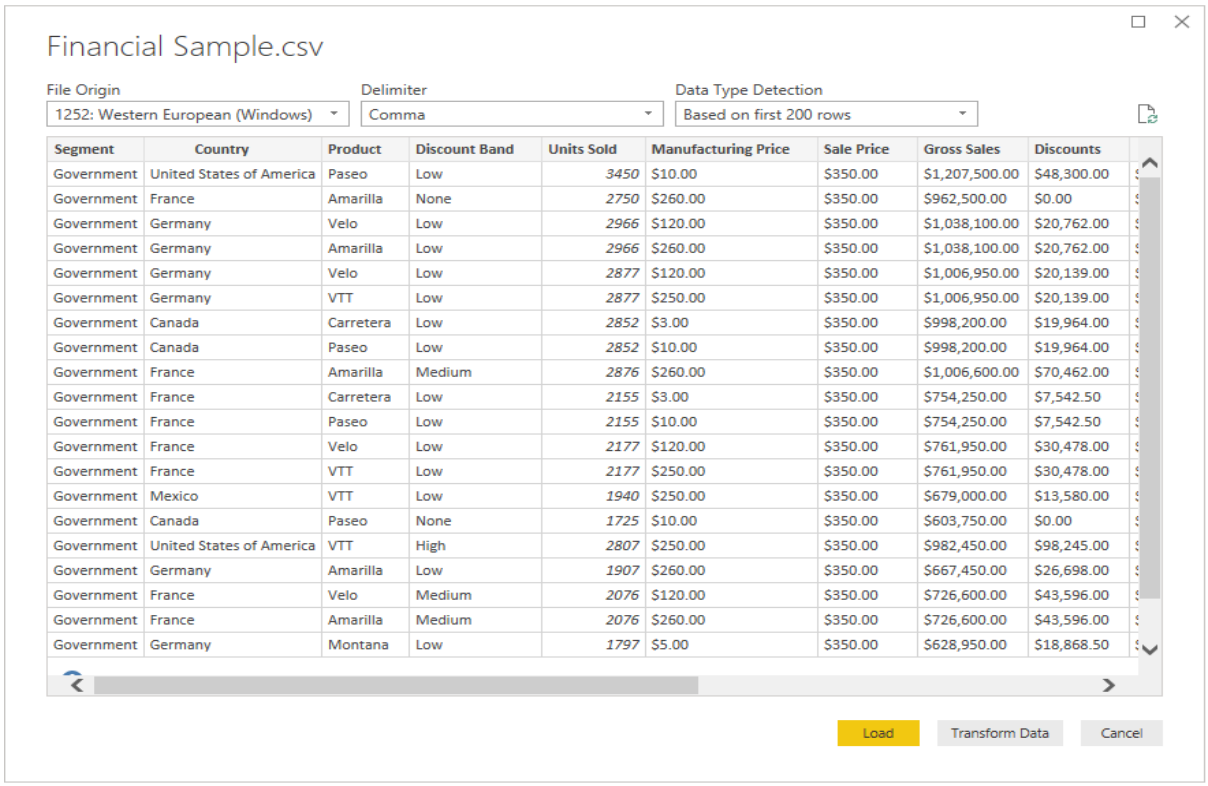
1) Open Power Bi Desktop
2) Click Get Data → Select Excel



3) Browse the Excel File and Click on Open

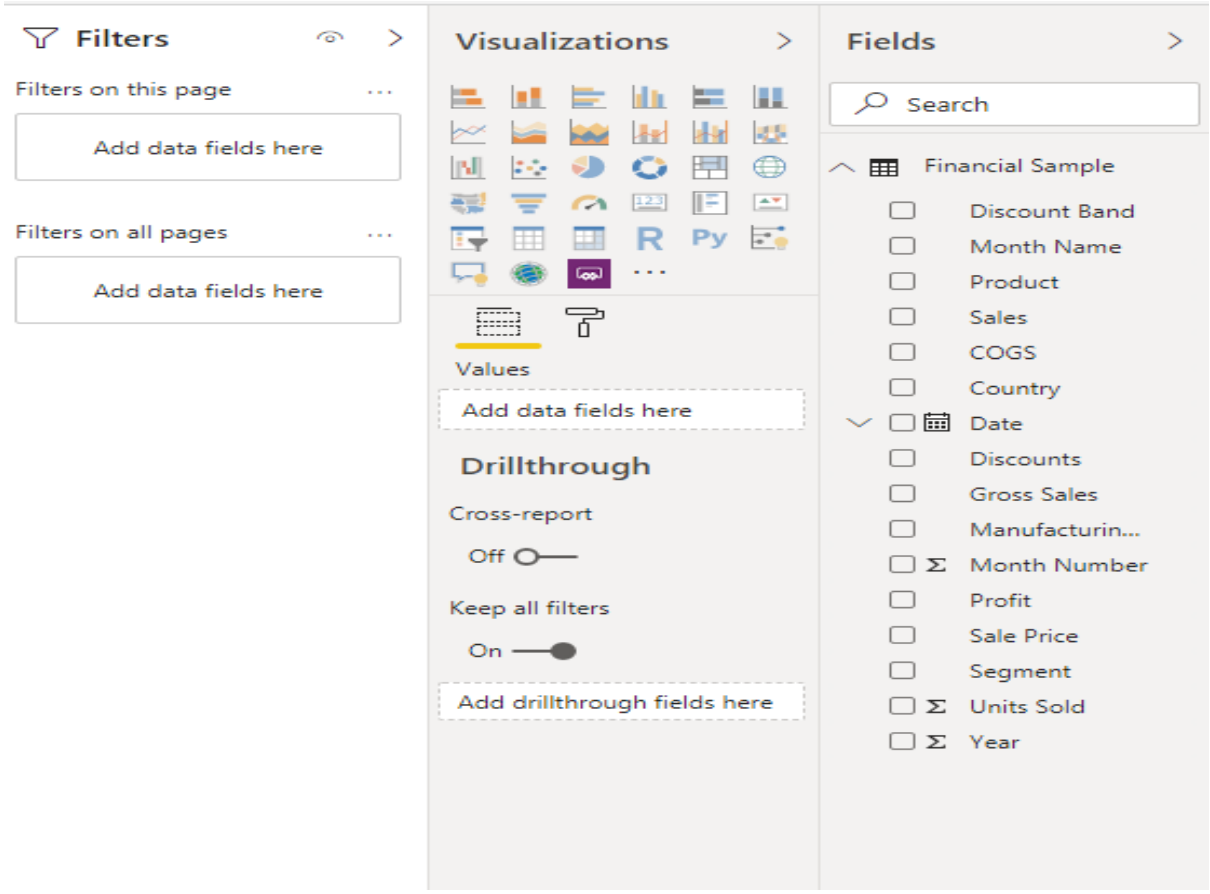


4) After Opening the Excel file Click on the Load Button

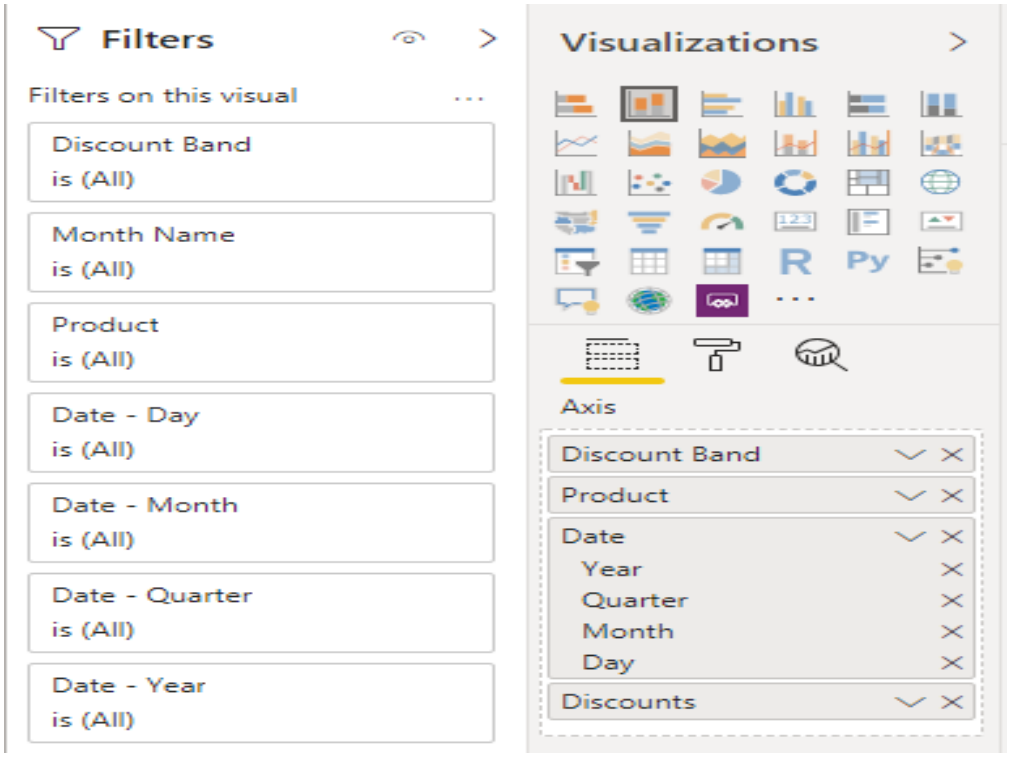


Segment	Country	Product	Discount Band	Units Sold	Manufacturing Price	Sale Price	Gross Sales	Discounts
Government	United States of America	Paseo	Low	3450	$10.00	$350.00	$1,207,500.00	$48,300.00
Government	France	Amarilla	None	2750	$260.00	$350.00	$962,500.00	$0.00
Government	Germany	Velo	Low	2966	$120.00	$350.00	$1,038,100.00	$20,762.00
Government	Germany	Amarilla	Low	2966	$260.00	$350.00	$1,038,100.00	$20,762.00
Government	Germany	Velo	Low	2877	$120.00	$350.00	$1,006,950.00	$20,139.00
Government	Germany	VTT	Low	2877	$250.00	$350.00	$1,006,950.00	$20,139.00
Government	Canada	Carretera	Low	2852	$3.00	$350.00	$998,200.00	$19,964.00
Government	Canada	Paseo	Low	2852	$10.00	$350.00	$998,200.00	$19,964.00
Government	France	Amarilla	Medium	2876	$260.00	$350.00	$1,006,600.00	$70,462.00
Government	France	Carretera	Low	2155	$3.00	$350.00	$754,250.00	$7,542.50
Government	France	Paseo	Low	2155	$10.00	$350.00	$754,250.00	$7,542.50
Government	France	Velo	Low	2177	$120.00	$350.00	$761,950.00	$30,478.00
Government	France	VTT	Low	2177	$250.00	$350.00	$761,950.00	$30,478.00
Government	Mexico	VTT	Low	1940	$250.00	$350.00	$679,000.00	$13,580.00
Government	Canada	Paseo	None	1725	$10.00	$350.00	$603,750.00	$0.00
Government	United States of America	VTT	High	2807	$250.00	$350.00	$982,450.00	$98,245.00
Government	Germany	Amarilla	Low	1907	$260.00	$350.00	$667,450.00	$26,698.00
Government	France	Velo	Medium	2076	$120.00	$350.00	$726,600.00	$43,596.00
Government	France	Amarilla	Medium	2076	$260.00	$350.00	$726,600.00	$43,596.00
Government	Germany	Montana	Low	1797	$5.00	$350.00	$628,950.00	$18,868.50

5) The Excel Sheet will get loaded, and you can find the contents of the excel sheet in the Fields Column.

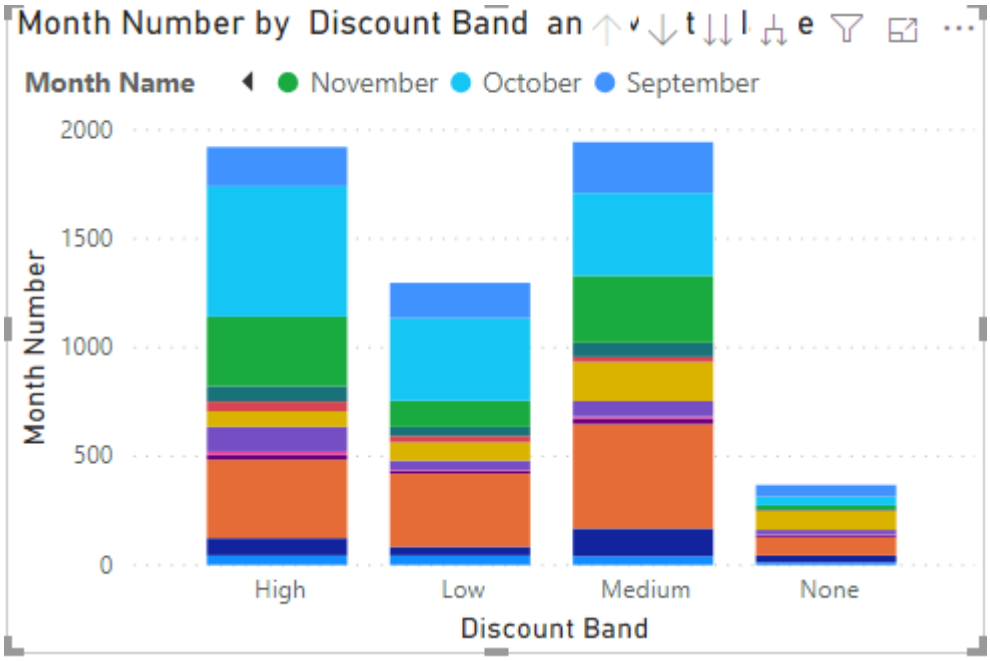


6) Go to the visualizations column to get a layout of the reports

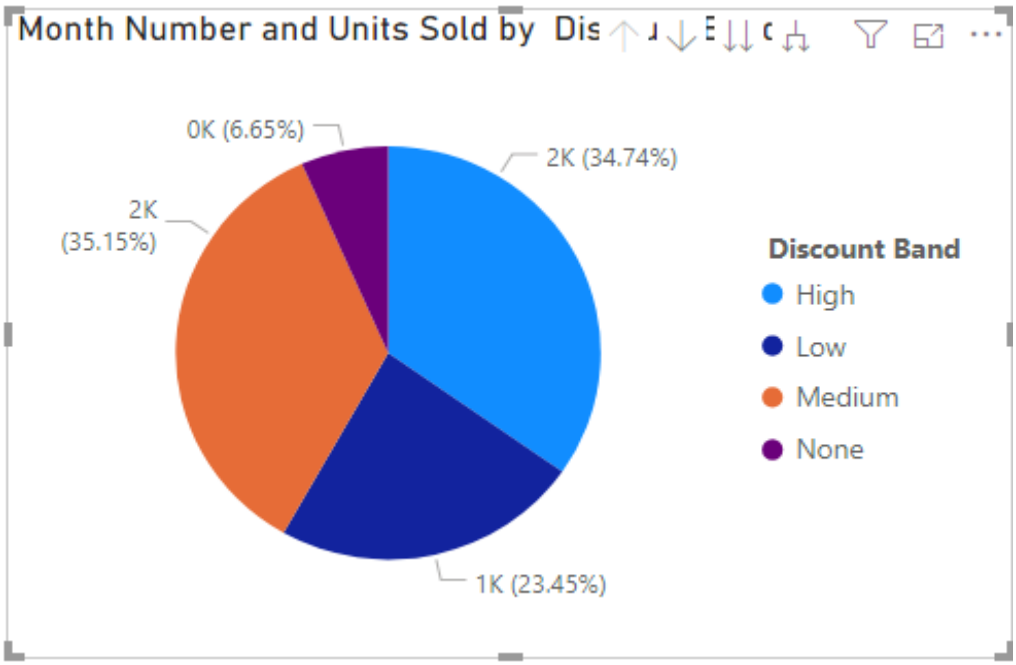


7) In the Field Column, Select the fields you want.

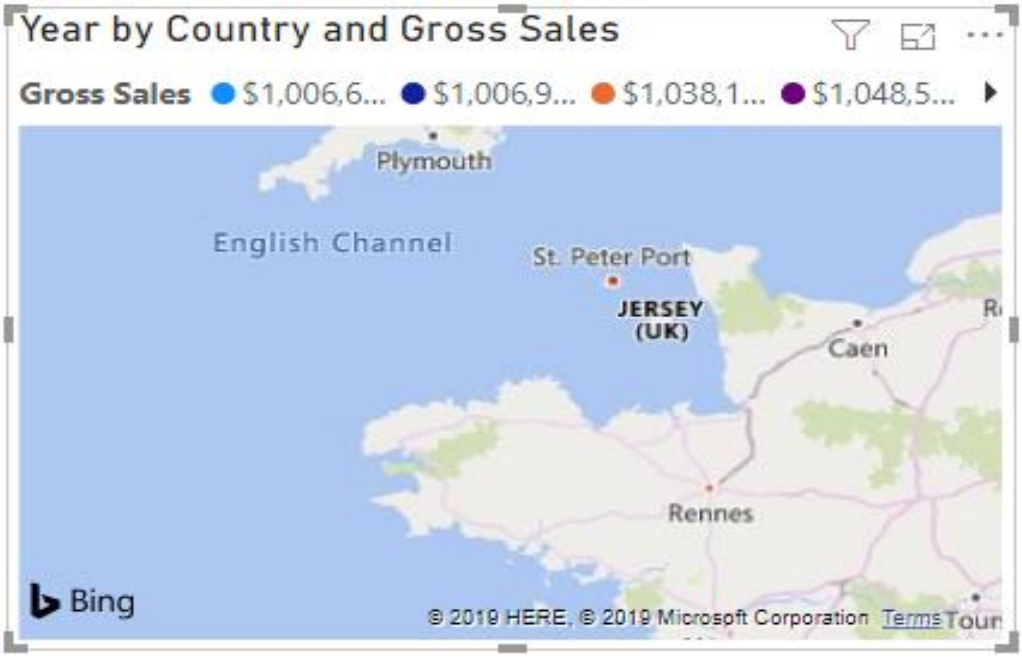
→Select Discount Band, Month name, Product, Date and Month Number.



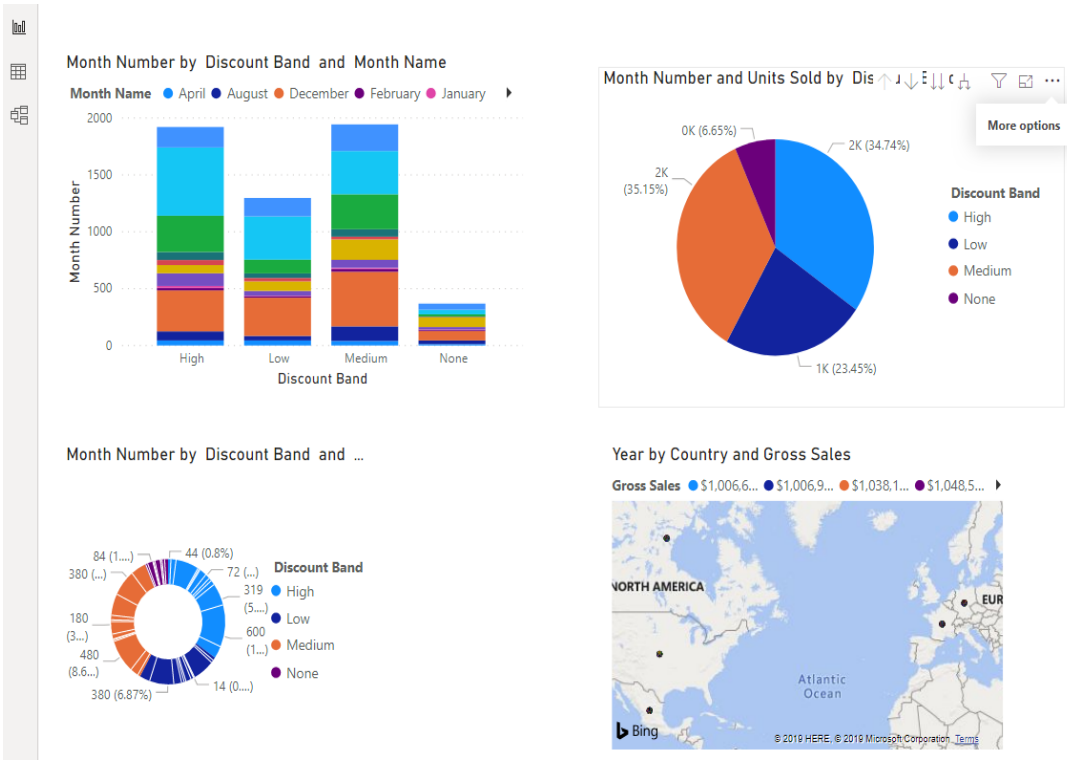
→Select Discount band, Product, Date, Discount, and Units Sold.

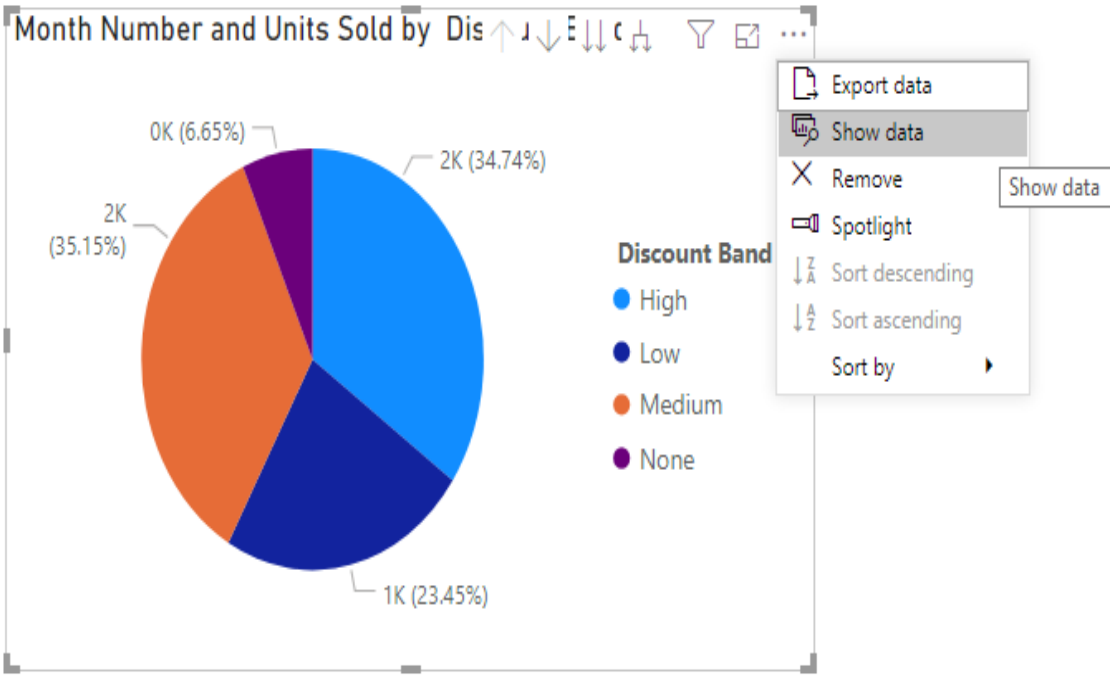


8) If you wand to display the country select on the map option in the visualization column and select the fields you want to display.

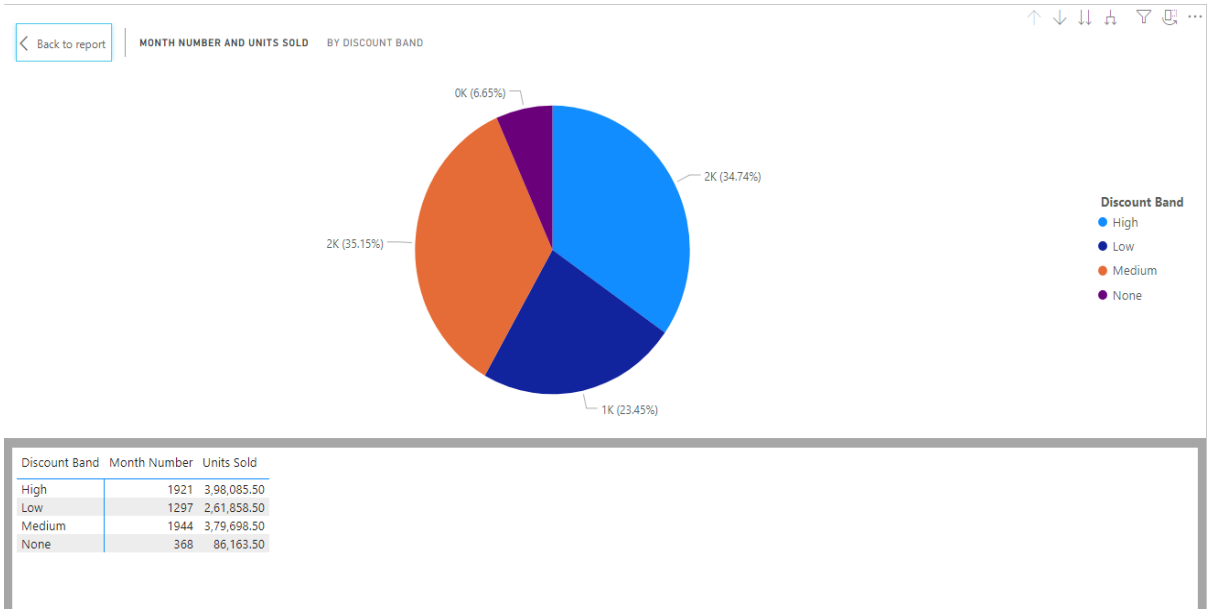


➔ Select any graph that you have made, Click on the upper corner on the 3 dots(…) and click on more options.



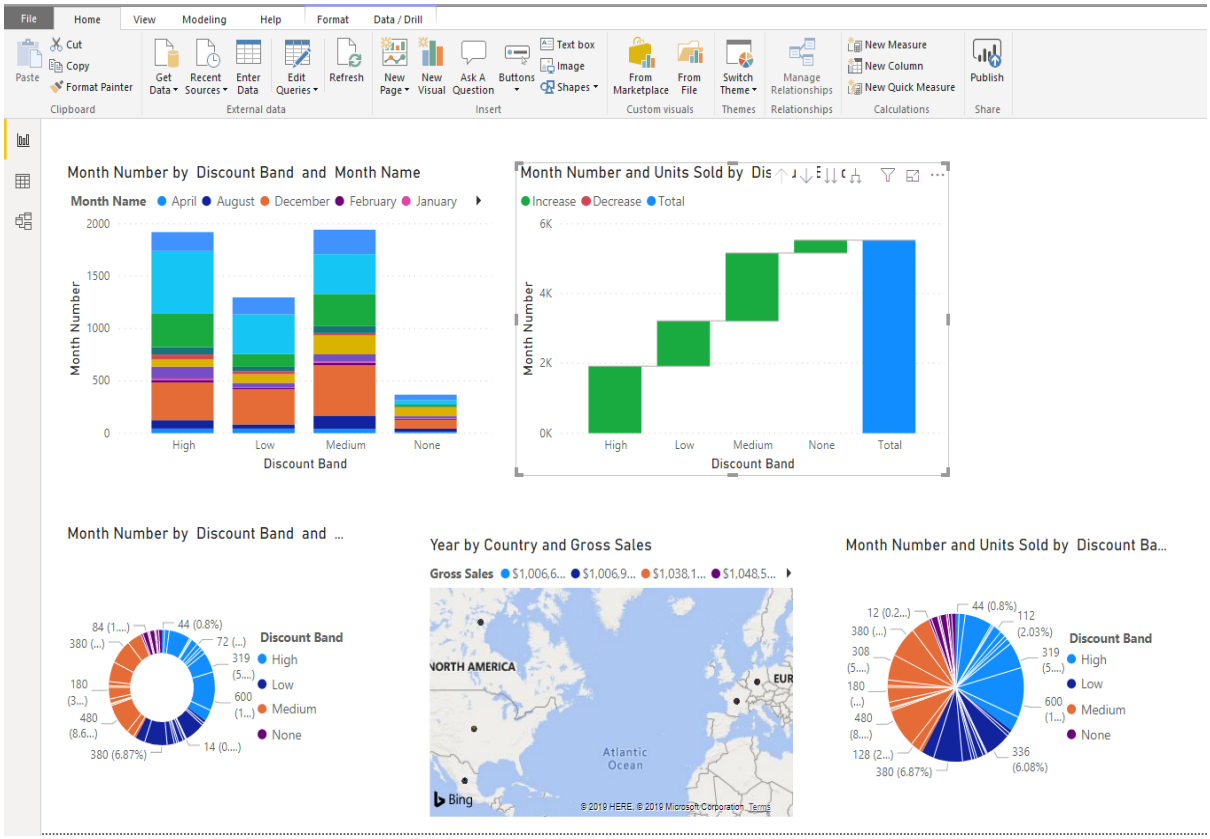


→The report will be displayed

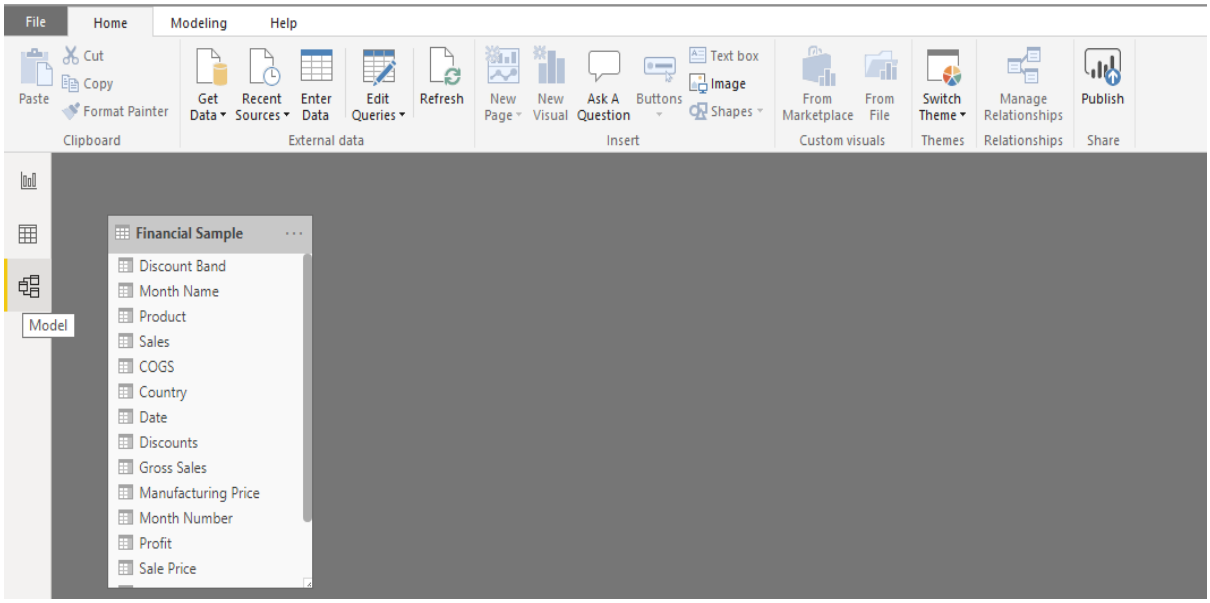


Discount Band	Month Number	Units Sold
High	1921	3,98,085.50
Low	1297	2,61,858.50
Medium	1944	3,79,698.50
None	368	86,163.50

9) The Final report will br displayed as follows:



10) To show the relationship, Click on Model tab



11) To Manage Relationships, Again Go to Get Data and Browse the Excel file