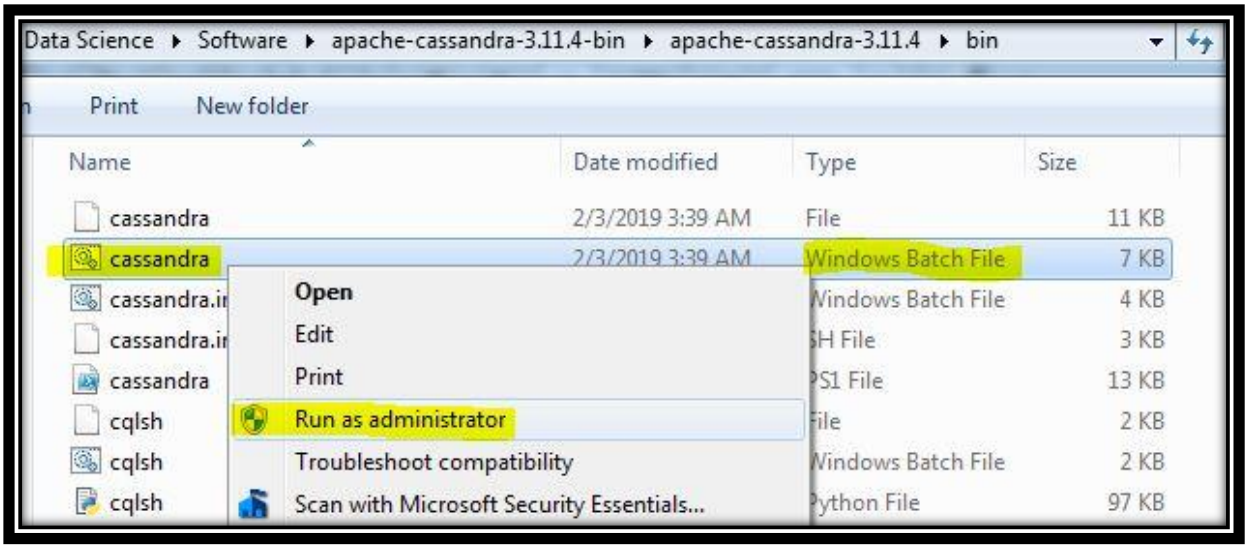
Practical 1
Cassandra Data Model

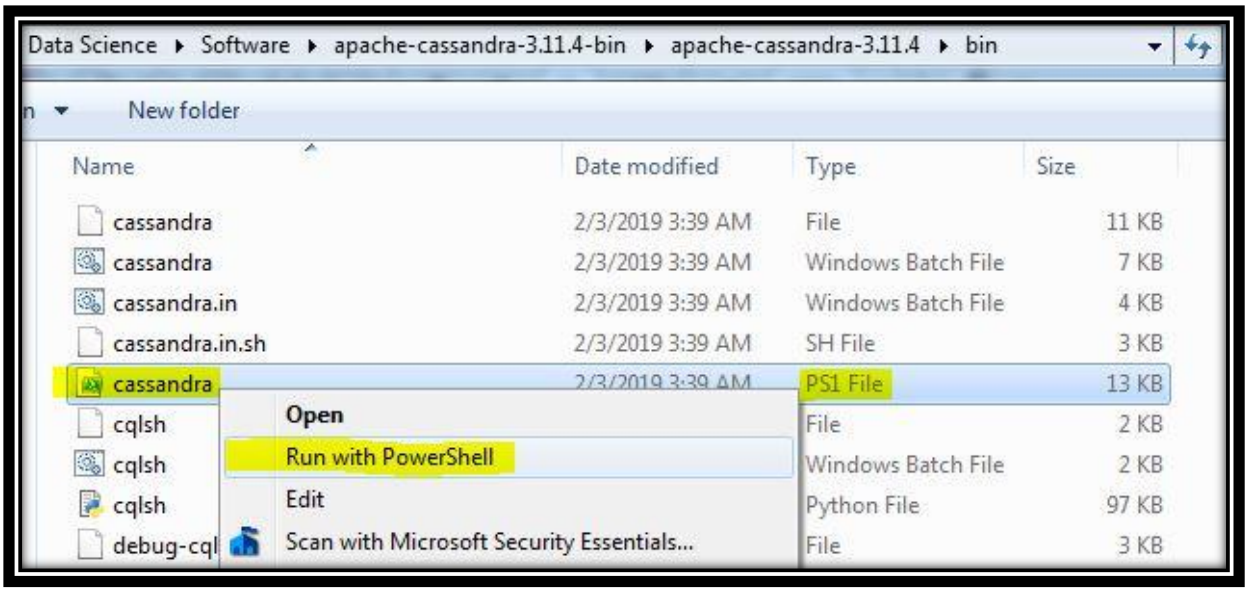
Step 1:
Go to Cassandra directory: C:\apache-cassandra-3.11.4\bin

Step 2:
Run Cassandra.bat file



NOTE: If you get an error on command prompt

Step 3:
Then **Run** Cassandra on PowerShell



Step 4:
Open C:\apache-cassandra-3.11.4\bin**cqlsh.py** with **python 2.7** and r**un**

Step 5:
Creating a Keyspace using Cqlsh

Code:

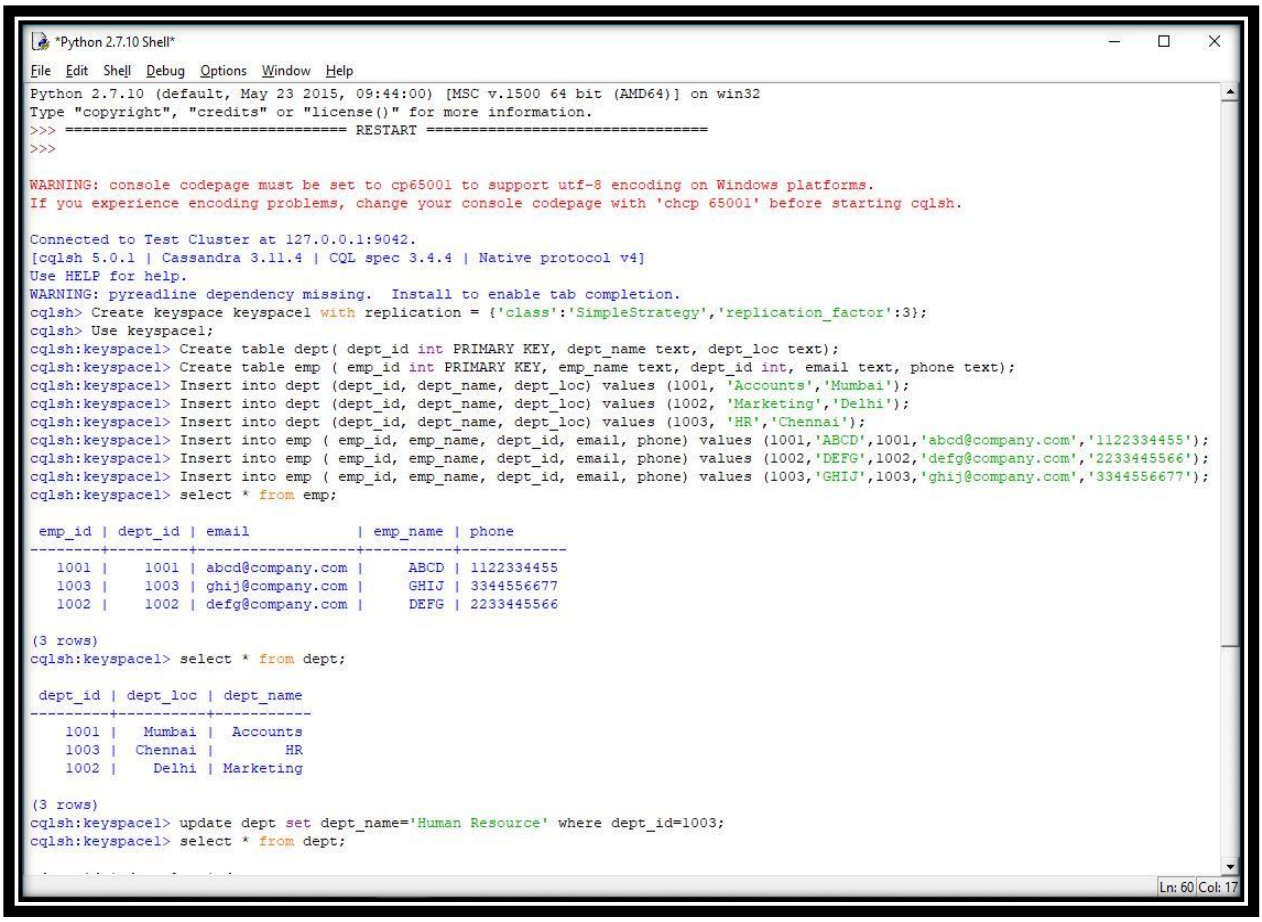
Create keyspace keyspace1 with replication = {'class':'SimpleStrategy','replication_factor': 3};

Use keyspace1;

Create table **dept** (dept_id int PRIMARY KEY, dept_name text, dept_loc text);

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

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

Insert into emp ( emp_id, emp_name, dept_id, email, phone )
values (1001, 'ABCD',1001, 'abcd@company.com',
'1122334455');
Insert into emp ( emp_id, emp_name, dept_id, email, phone )
values (1002, 'DEFG',1002, 'defg@company.com',
'2233445566');
Insert into emp ( emp_id, emp_name, dept_id, email, phone )
values (1003, 'GHIJ',1003, 'ghij@company.com',
'3344556677');


select * from emp;

select * from dept;

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

delete from emp where emp_id=1006;

select * from emp;
```



```
*Python 2.7.10 Shell*
File Edit Shell Debug Options Window Help
Python 2.7.10 (default, May 23 2015, 09:44:00) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cqlsh> Create keyspace keyspace1 with replication = {'class':'SimpleStrategy','replication_factor':3};
cqlsh> Use keyspace1;
cqlsh:keyspace1> Create table dept( dept_id int PRIMARY KEY, dept_name text, dept_loc text);
cqlsh:keyspace1> Create table emp ( emp_id int PRIMARY KEY, emp_name text, dept_id int, email text, phone text);
cqlsh:keyspace1> Insert into dept (dept_id, dept_name, dept_loc) values (1001, 'Accounts','Mumbai');
cqlsh:keyspace1> Insert into dept (dept_id, dept_name, dept_loc) values (1002, 'Marketing','Delhi');
cqlsh:keyspace1> Insert into dept (dept_id, dept_name, dept_loc) values (1003, 'HR','Chennai');
cqlsh:keyspace1> Insert into emp ( emp_id, emp_name, dept_id, email, phone) values (1001,'ABCD',1001,'abcd@company.com','1122334455');
cqlsh:keyspace1> Insert into emp ( emp_id, emp_name, dept_id, email, phone) values (1002,'DEFG',1002,'defg@company.com','2233445566');
cqlsh:keyspace1> Insert into emp ( emp_id, emp_name, dept_id, email, phone) values (1003,'GHIJ',1003,'ghij@company.com','3344556677');
cqlsh:keyspace1> select * from emp;

 emp_id | dept_id | email            | emp_name | phone
--------+---------+------------------+----------+------------
   1001 |    1001 | abcd@company.com |     ABCD | 1122334455
   1003 |    1003 | ghij@company.com |     GHIJ | 3344556677
   1002 |    1002 | defg@company.com |     DEFG | 2233445566

(3 rows)
cqlsh:keyspace1> select * from dept;

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

(3 rows)
cqlsh:keyspace1> update dept set dept_name='Human Resource' where dept_id=1003;
cqlsh:keyspace1> select * from dept;
                                                                    Ln: 60 Col: 17
```

```
*Python 2.7.10 Shell*
File Edit Shell Debug Options Window Help
 dept_id | dept_loc | dept_name
---------+----------+----------------
    1001 |   Mumbai |       Accounts
    1003 |  Chennai | Human Resource
    1002 |    Delhi |      Marketing

(3 rows)
cqlsh:keyspace1> delete from emp where emp_id=1003;
cqlsh:keyspace1> select * from emp;

 emp_id | dept_id | email            | emp_name | phone
--------+---------+------------------+----------+------------
   1001 |    1001 | abcd@company.com |     ABCD | 1122334455
   1002 |    1002 | defg@company.com |     DEFG | 2233445566

(2 rows)
cqlsh:keyspace1>
                                                                    Ln: 60 Col: 17
```

Practical 2

Homogeneous Ontology for Recursive Uniform Schema

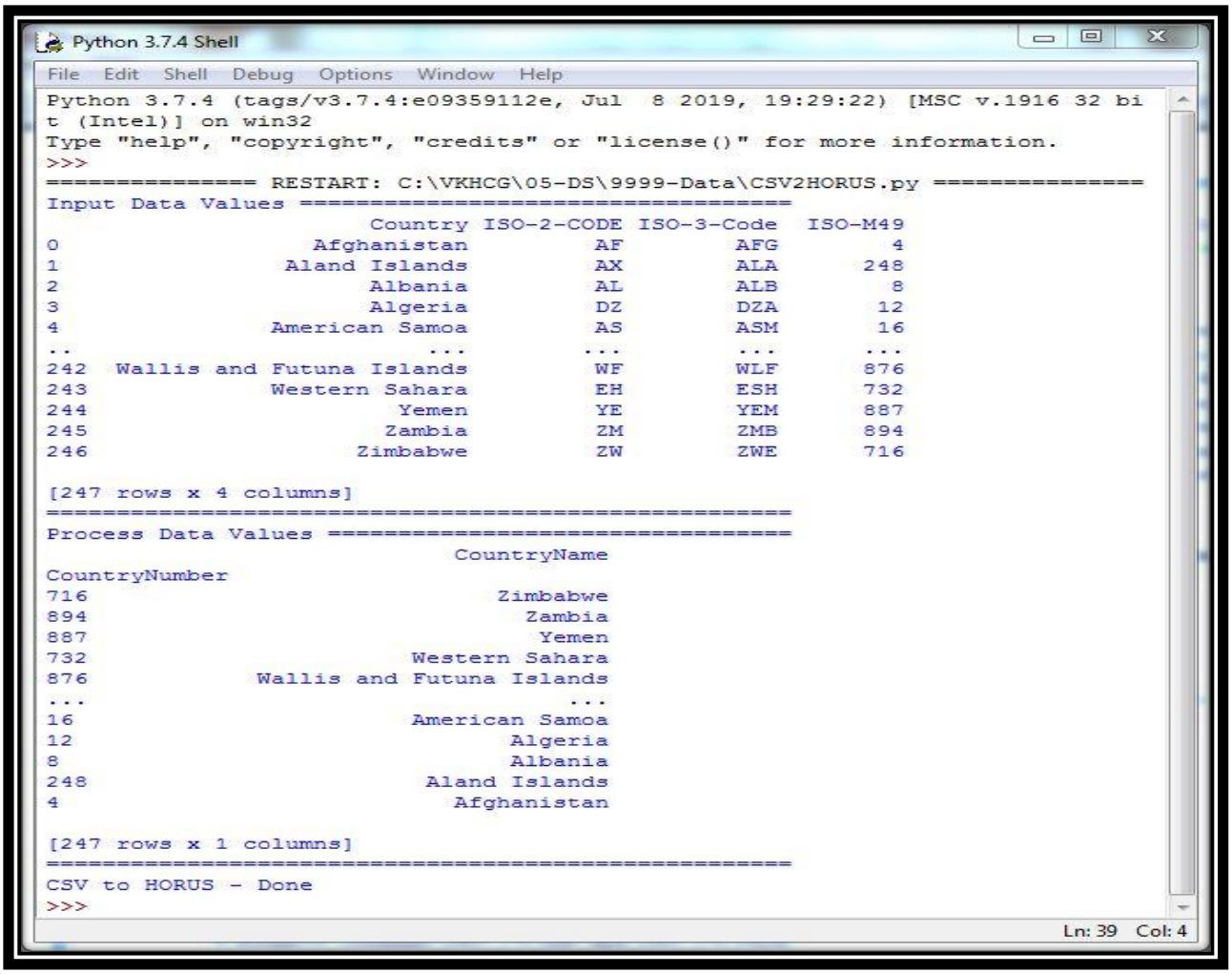
A. Text delimited CSV to HORUS format.

Code:

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

# Standard Tools

#============================================================

import pandas as pd

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

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

InputData=pd.read_csv(sInputFileName,encoding="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='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'

OutputData.to_csv(sOutputFileName, index = False)

print('CSV to HORUS - Done')

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

Output:



B. XML to HORUS Format Code:

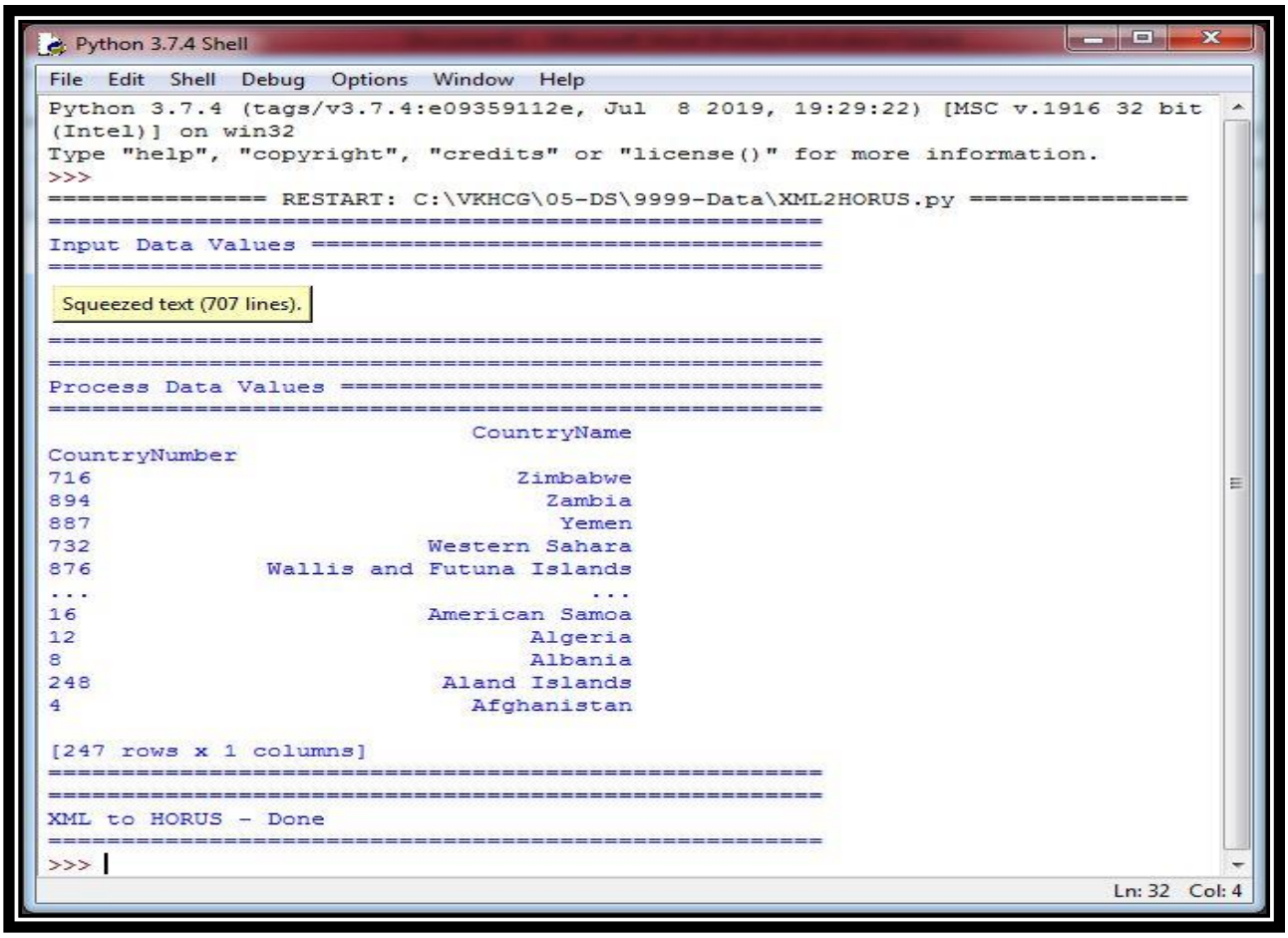
```
# Utility Start XML to HORUS ================================

# Standard Tools

#============================================================

import pandas as pd

import xml.etree.ElementTree as ET

#============================================================

def df2xml(data):

    header = data.columns

    root = ET.Element('root')
```

```
    for row in range(data.shape[0]):
        entry = ET.SubElement(root,'entry')
        for index in range(data.shape[1]):
            schild=str(header[index])
            child = ET.SubElement(entry, schild)
            if str(data[schild][row]) != 'nan':
                child.text = str(data[schild][row])
            else:
                child.text = 'n/a'
            entry.append(child)
    result = ET.tostring(root)
    return result
#=============================================================
def xml2df(xml_data):
    root = ET.XML(xml_data)
    all_records = []
    for i, child in enumerate(root):
        record = {}
        for subchild in child:
            record[subchild.tag] = subchild.text
        all_records.append(record)
    return pd.DataFrame(all_records)
#=============================================================
```

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

#==========================================================

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

InputData = open(sInputFileName).read()

print('==================================================')

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

print('==================================================')

print(InputData)

print('==================================================')

#==========================================================

# Processing Rules ========================================

#==========================================================

ProcessDataXML=InputData

# XML to Data Frame

ProcessData=xml2df(ProcessDataXML)

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

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

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

# Rename Country and ISO-M49

ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)

ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

# Set new Index

ProcessData.set_index('CountryNumber', inplace=True)
```

```
# Sort data by CurrencyNumber

ProcessData.sort_values('CountryName', axis=0, ascending=False,
inplace=True)

print('================================================')

print('Process Data Values ================================')

print('================================================')

print(ProcessData)

print('================================================')

#============================================================

# Output Agreement ========================================

#============================================================

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'

OutputData.to_csv(sOutputFileName, index = False)

print('================================================')

print('XML to HORUS - Done')

print('================================================')

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

Output:



C. JSON to HORUS Format

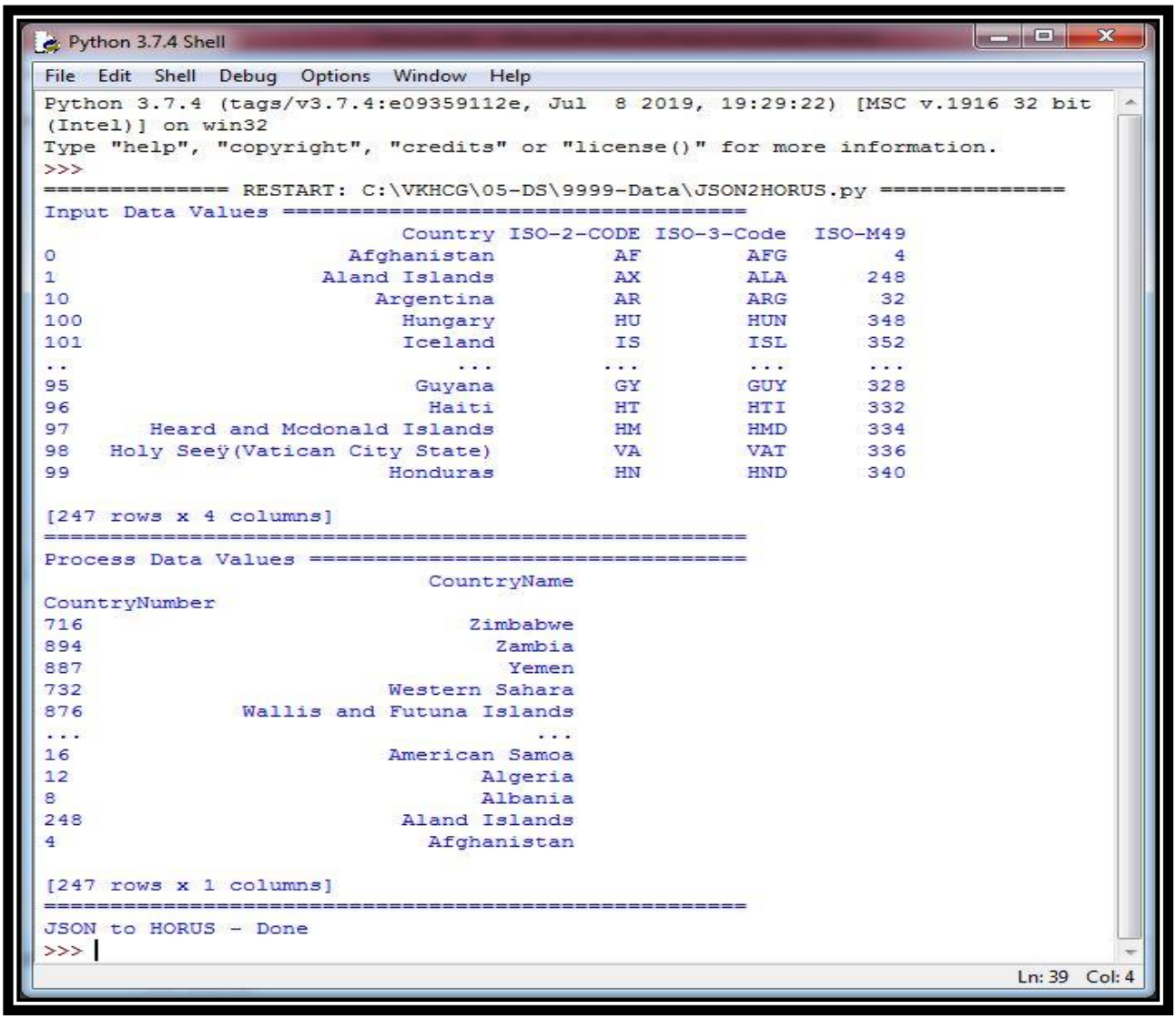
Code:

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

# Standard Tools

#=============================================================

import pandas as pd

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

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

InputData=pd.read_json(sInputFileName,

                       orient='index',

                       encoding="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='C:/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('JSON to HORUS - Done')
```

Utility done ==========

Output:



D. MySql Database to HORUS Format Code:

Utility Start Database to HORUS ==========

Standard Tools

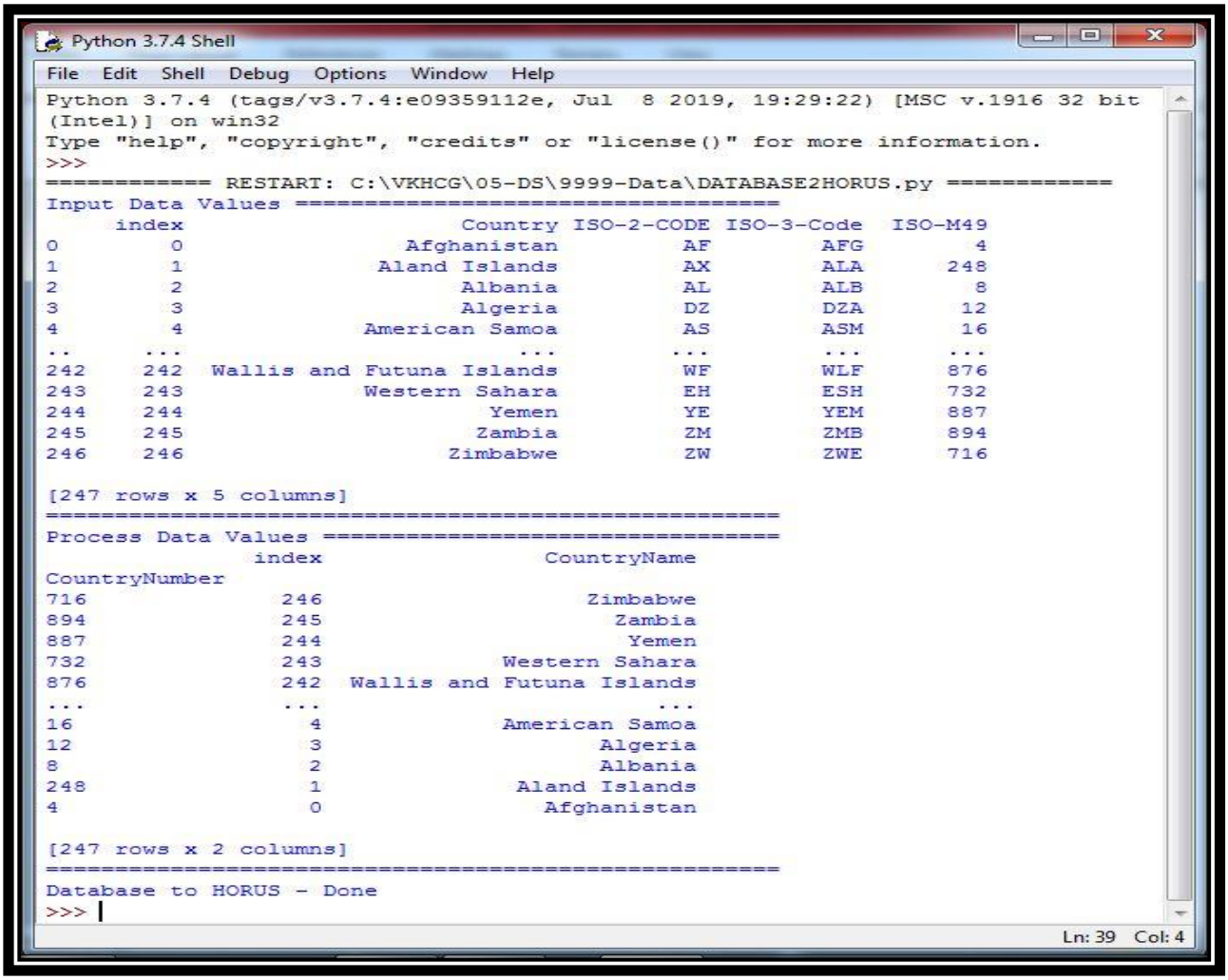
#==========

import pandas as pd

```
import sqlite3 as sq
# Input Agreement ==========================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)
print('Input Data Values ===================================')
print(InputData)
print('=================================================')
# Processing Rules ========================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False,
inplace=True)
print('Process Data Values ================================')
```

```
print(ProcessData)

print('================================================')

# Output Agreement ============================================

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'

OutputData.to_csv(sOutputFileName, index = False)

print('Database to HORUS - Done')

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

Output:



E. Picture (JPEG) to HORUS Format

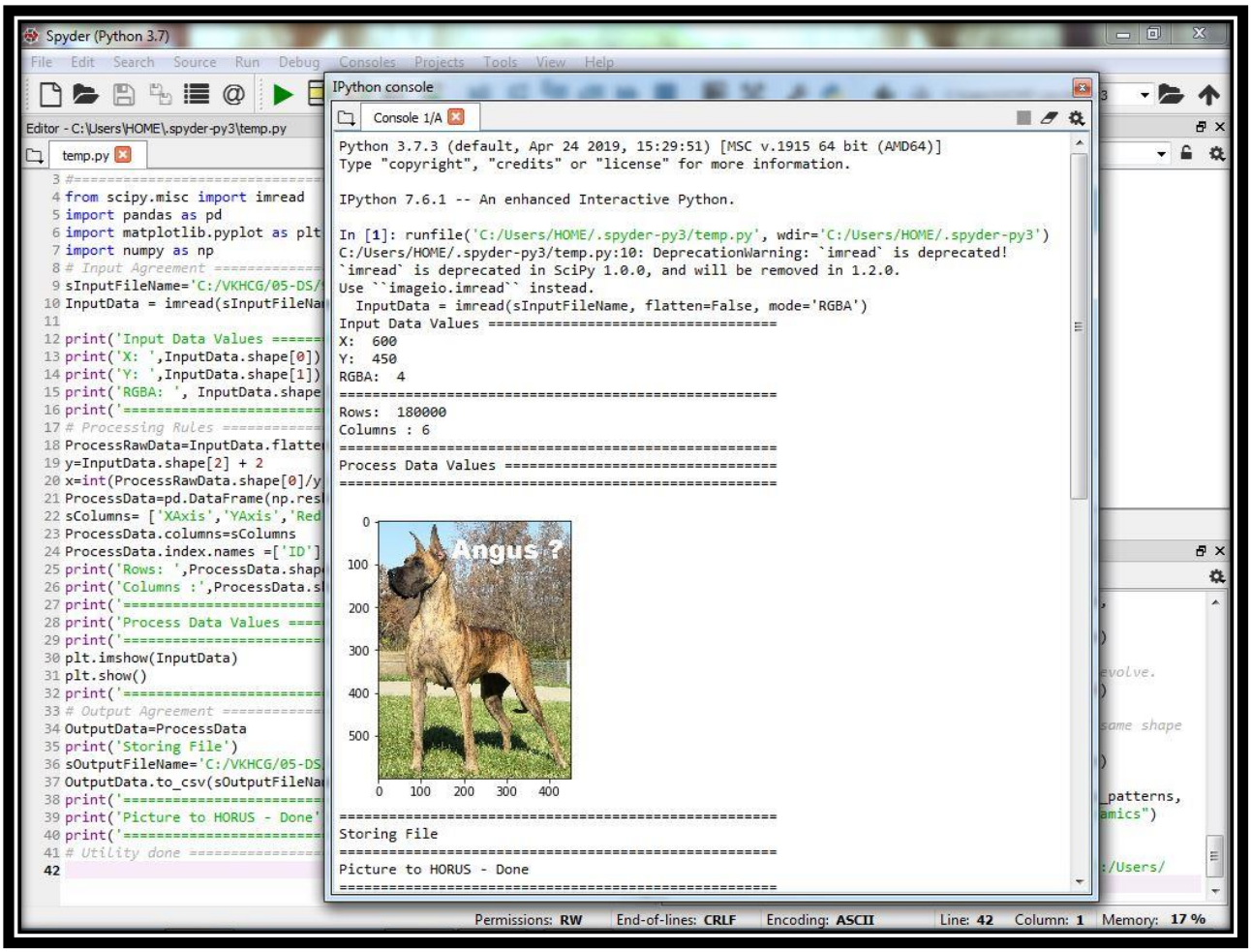
(Use SPYDER to run this program)

Code:

```
# Utility Start Picture to HORUS ================================

# Standard Tools

#============================================================

from scipy.misc import imread

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

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

sInputFileName='C:/VKHCG/05-DS/9999-Data/Angus.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('==================================================')
print('Process Data Values ==============================')
print('==================================================')
plt.imshow(InputData)
plt.show()
print('==================================================')
# Output Agreement ========================================
OutputData=ProcessData
print('Storing File')
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('==================================================')
print('Picture to HORUS - Done')
print('==================================================')
# Utility done ============================================
```

Output:



F. Video to HORUS Format
Movie to Frames

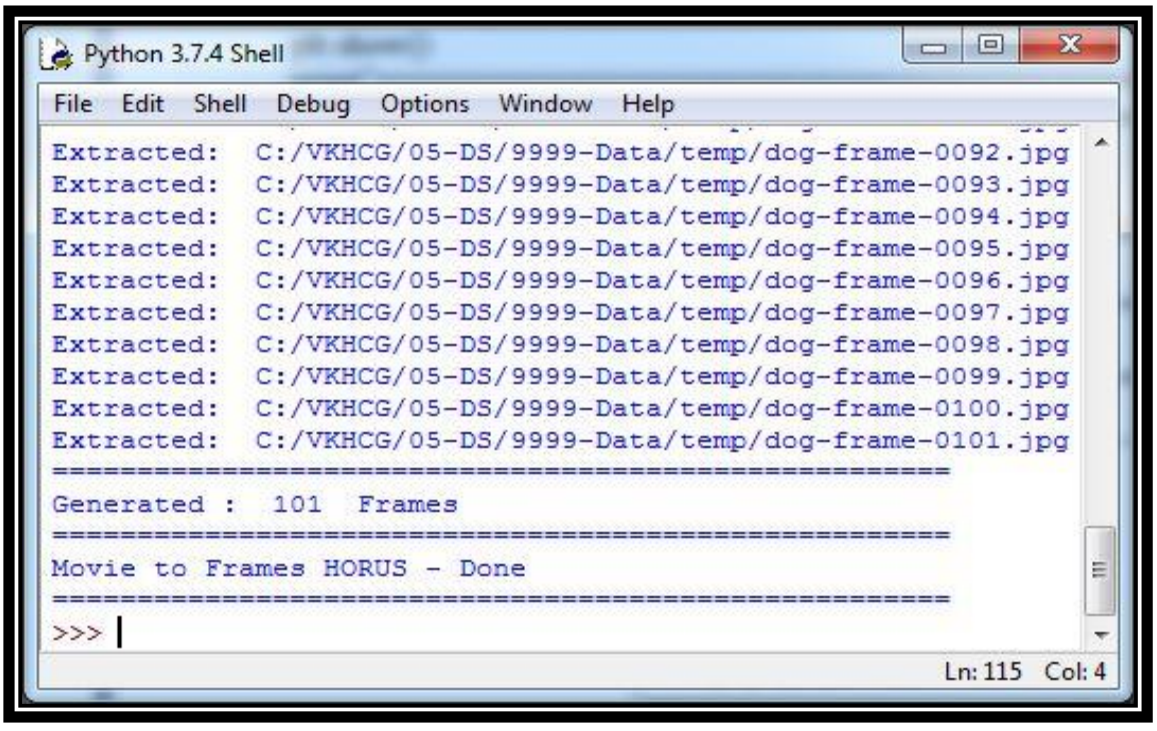
Code:

```
# Utility Start Movie to HORUS (Part 1) ======================

# Standard Tools

#============================================================

import os

import shutil

import cv2

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

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

```
    if count > 100: # exit
        break
    count += 1
print('=====================================================')
print('Generated : ', count, ' Frames')
print('=====================================================')
print('Movie to Frames HORUS - Done')
print('=====================================================')
# Utility done ==============================================
```

Output:



Now frames are created and need to load them into HORUS.

Frames to Horus (Use SPYDER to run this program)

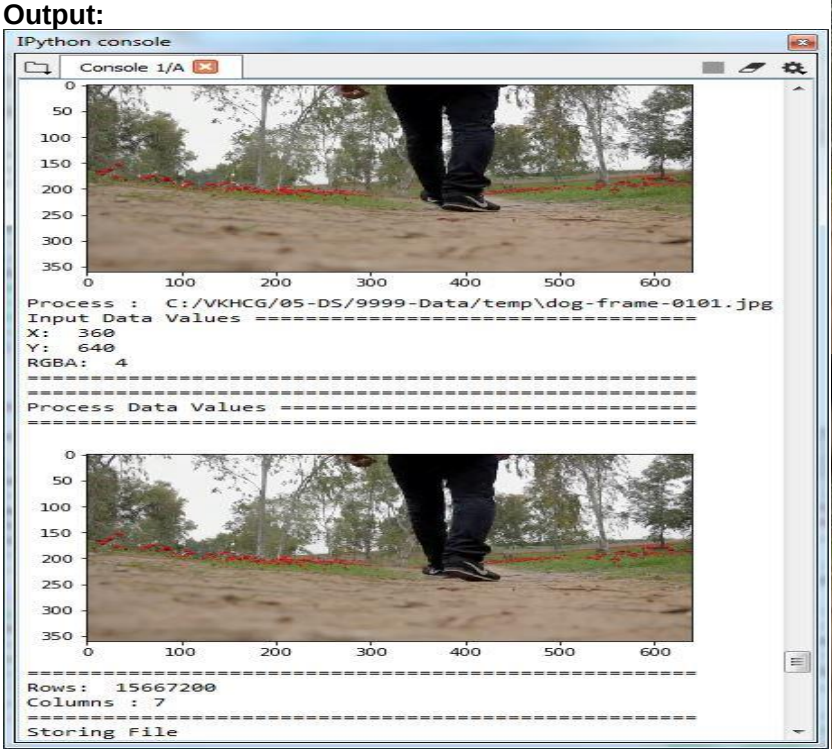
Code:

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

        InputData = imread(sInputFileName, 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)
        ProcessFrameData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
        ProcessFrameData['Frame']=file
        print('================================================')
        print('Process Data Values ============================')
        print('================================================')
        plt.imshow(InputData)
        plt.show()

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

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

Output:



Check the files from C:\VKHCG\05-DS\9999-Data\temp

The movie clip is converted into 102 picture frames and then to HORUS format.

G. Audio to HORUS Format
Code:

```
# Utility Start Audio to HORUS ==============================

# Standard Tools

#============================================================
=======

from scipy.io import wavfile

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#============================================================
=======

def show_info(aname, a,r):

    print ('----------------')

    print ("Audio:", aname)

    print ('----------------')

    print ("Rate:", r)

    print ('----------------')

    print ("shape:", a.shape)

    print ("dtype:", a.dtype)

    print ("min, max:", a.min(), a.max())

    print ('----------------')
```

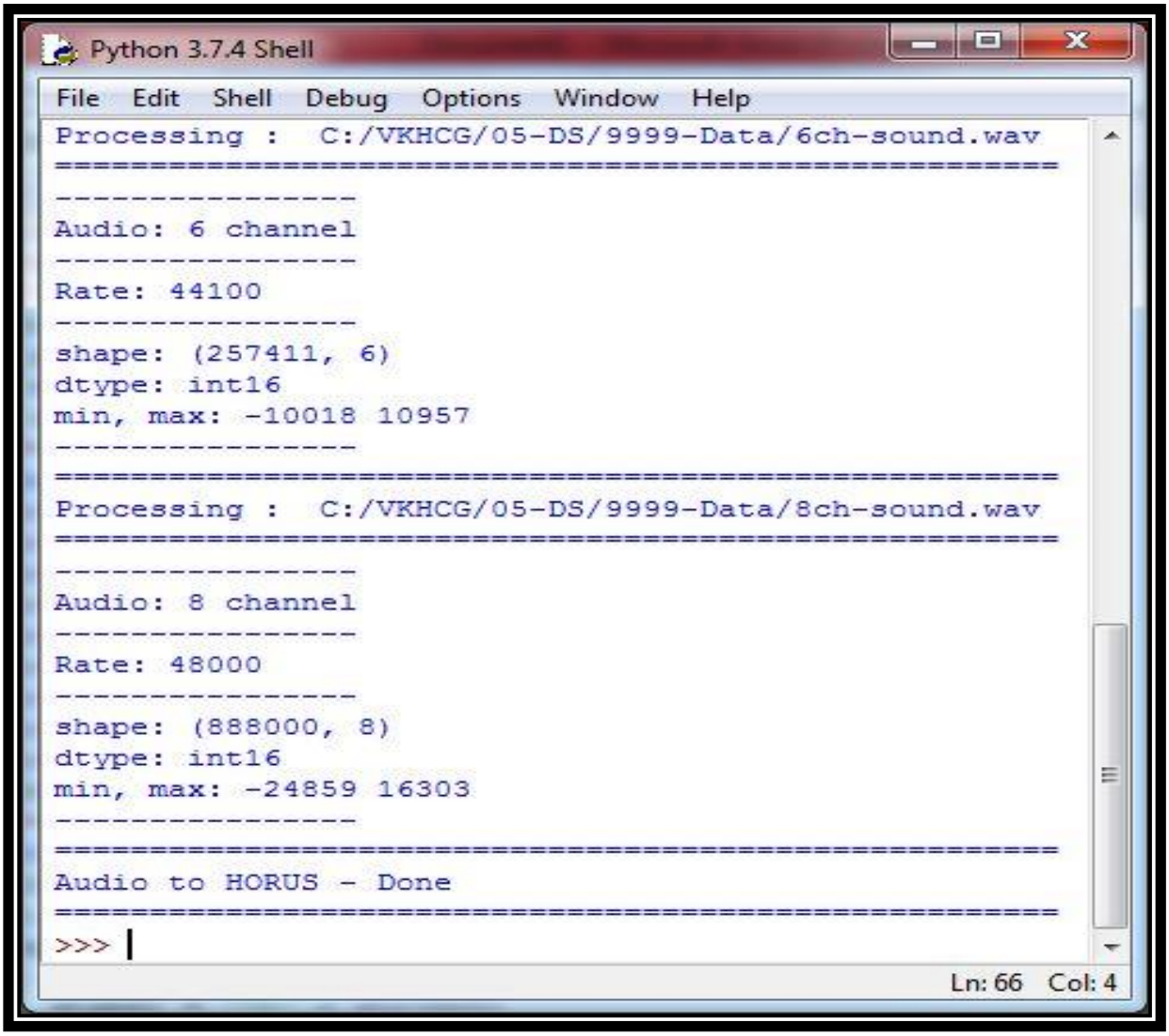
```
    plot_info(aname, a,r)
#================================================
=======
def plot_info(aname, a,r):
    sTitle= 'Signal Wave - '+ aname + ' at ' + str(r) + 'hz'
    plt.title(sTitle)
    sLegend=[]
    for c in range(a.shape[1]):
        sLabel = 'Ch' + str(c+1)
        sLegend=sLegend+[str(c+1)]
        plt.plot(a[:,c], label=sLabel)
    plt.legend(sLegend)
    plt.show()
#================================================
=======
sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'
print('================================================
===')
print('Processing : ', sInputFileName)
print('================================================
===')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
```

```
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#=================================================
=======
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('================================================
===')
print('Processing : ', sInputFileName)
print('================================================
===')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("4 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#=================================================
=======
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('================================================
===')
print('Processing : ', sInputFileName)
print('================================================
===')
```

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

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

print('================================================
===')

#================================================
=======

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

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

Output:



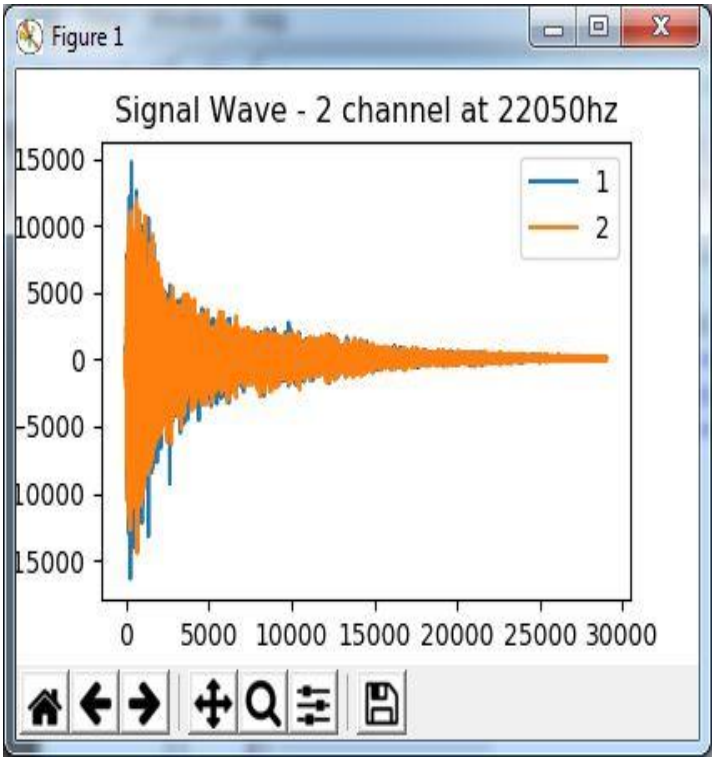


Figure 1
Signal Wave - 2 channel at 22050hz
15000
10000
5000
0
-5000
10000
15000
0 5000 10000 15000 20000 25000 30000
1
2

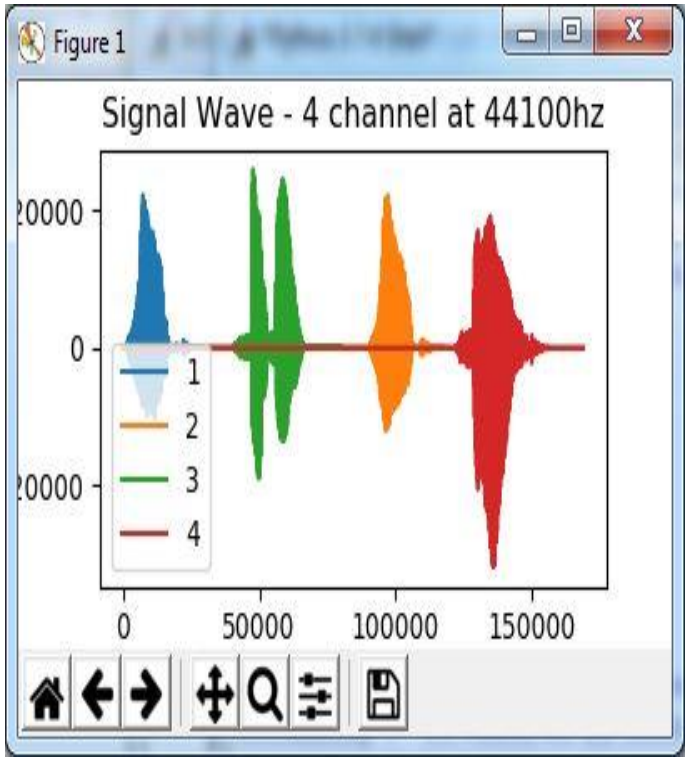


Figure 1
Signal Wave - 4 channel at 44100hz
20000
0
20000
0 50000 100000 150000
1
2
3
4

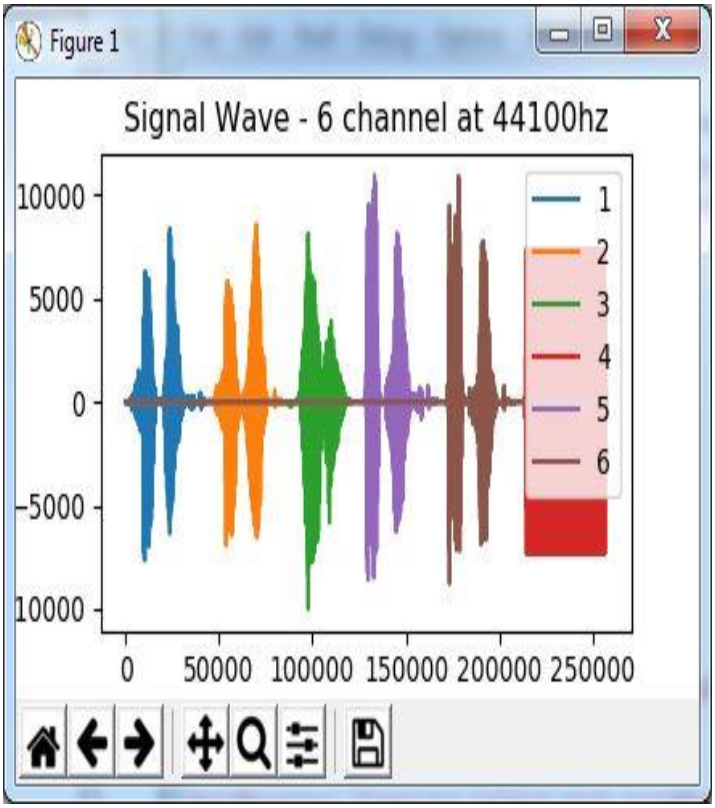


Figure 1
Signal Wave - 6 channel at 44100hz
10000
5000
0
-5000
10000
0 50000 100000 150000 200000 250000
1
2
3
4
5
6

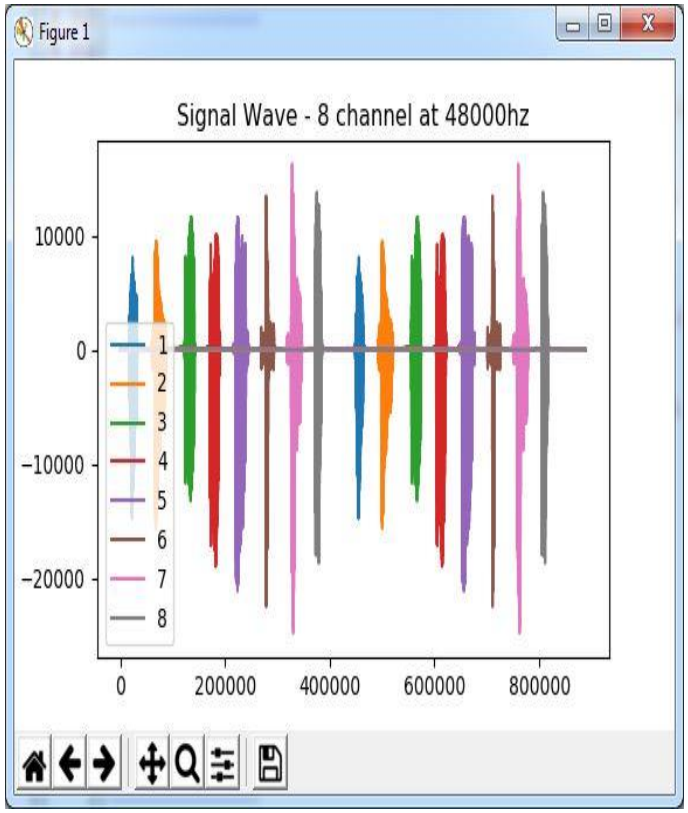


Figure 1
Signal Wave - 8 channel at 48000hz
10000
0
−10000
−20000
0 200000 400000 600000 800000
1
2
3
4
5
6
7
8

Practical 3

Utilities and Auditing

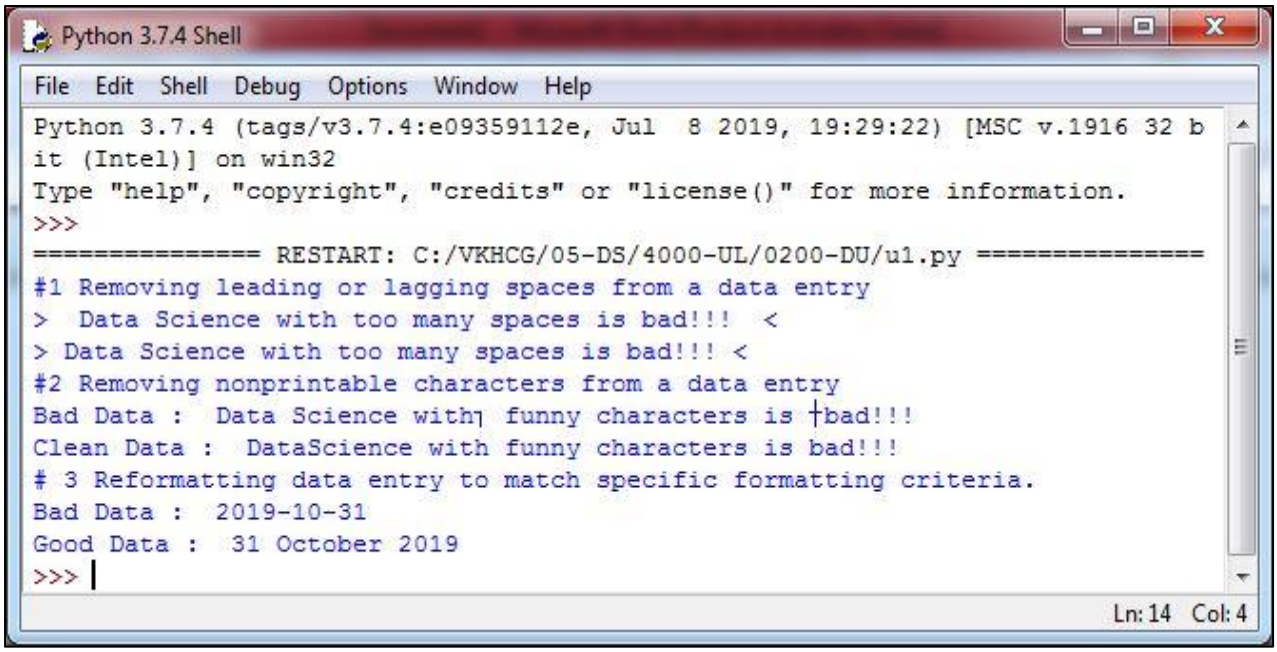
A. Fixers Utilities:

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

```
#--------------------------- Program to Demonstrate Fixers utilities -
-----------------

import string

import datetime as dt


# 1 Removing leading or lagging spaces from a data entry

print('#1 Removing leading or lagging spaces from a data entry');

baddata = " Data Science with too many spaces is bad!!! "

print('>',baddata,'<')

cleandata=baddata.strip()

print('>',cleandata,'<')


# 2 Removing nonprintable characters from a data entry

print('#2 Removing nonprintable characters from a data entry')

printable = set(string.printable)

baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!"

cleandata=''.join(filter(lambda x: x in string.printable,baddata))

print('Bad Data : ',baddata);

print('Clean Data : ',cleandata)
```

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

Output:



B. Data Binning or Bucketing

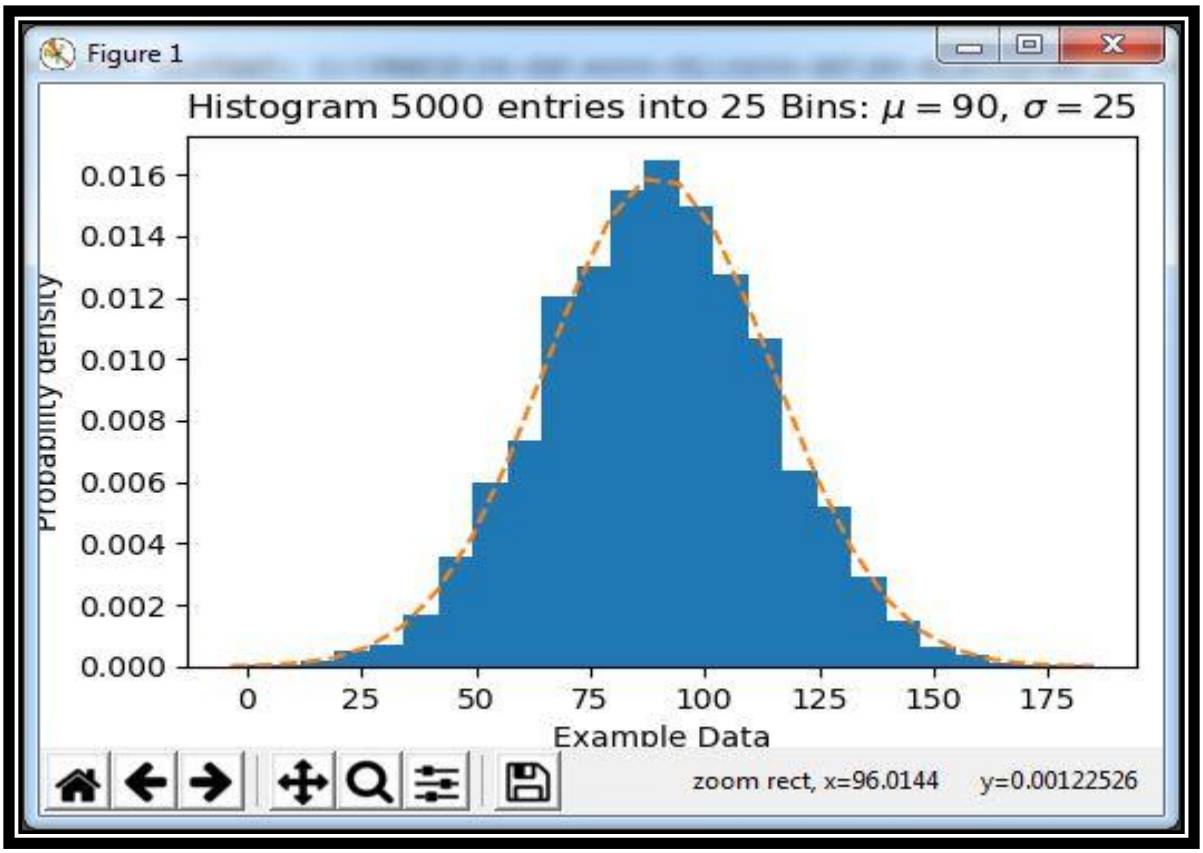
Code :

```
import numpy as np
```

```
import matplotlib.mlab as mlab

import matplotlib.pyplot as plt

import scipy.stats as stats #change

np.random.seed(0)

# example data

mu = 90  # mean of distribution

sigma = 25  # standard deviation of distribution

x = mu + sigma * np.random.randn(5000)

num_bins = 25

fig, ax = plt.subplots()

# the histogram of the data

n, bins, patches = ax.hist(x, num_bins, density=1)

# add a 'best fit' line

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

#y = mlab.normpdf(bins, mu, sigma) in 2.1

ax.plot(bins, y, '--')

ax.set_xlabel('Example Data')

ax.set_ylabel('Probability density')

sTitle=r'Histogram ' + str(len(x)) + ' entries into ' + str(num_bins) + ' Bins:
$\mu=' + str(mu) + '$, $\sigma=' + str(sigma) + '$'

ax.set_title(sTitle)

fig.tight_layout()

sPathFig='C:/VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png'

fig.savefig(sPathFig)
```

plt.show()

Output:



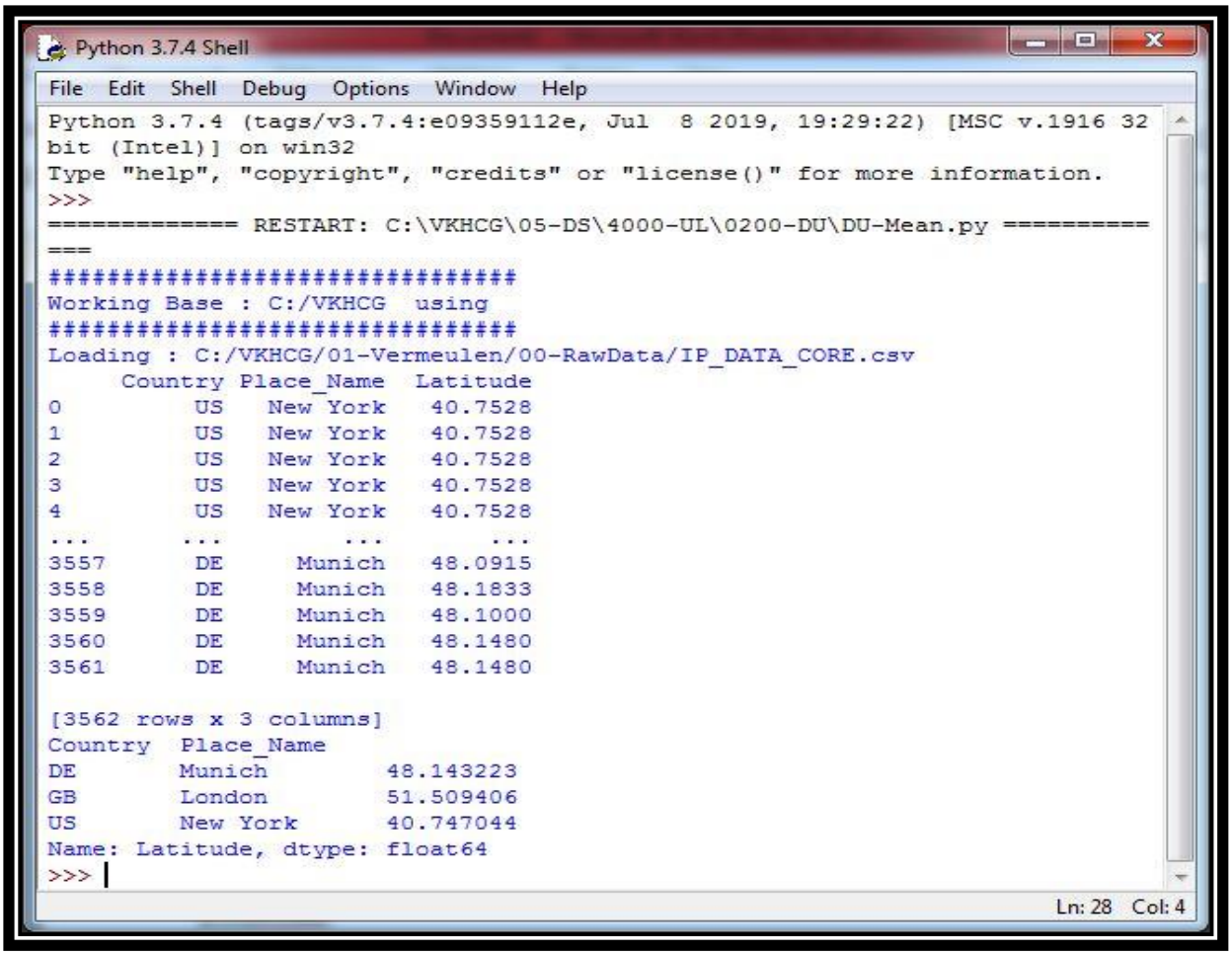
C. Averaging of Data

Code:

```
import pandas as pd

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

InputFileName='IP_DATA_CORE.csv'

OutputFileName='Retrieve_Router_Location.csv'

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

Base='C:/VKHCG'
```

```
print('###################################')
print('Working Base :',Base, ' using ')
print('###################################')
#####################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
AllData=IP_DATA_ALL[['Country', 'Place_Name','Latitude']]
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
print(MeanData)
```

Output:



D. Outlier Detection

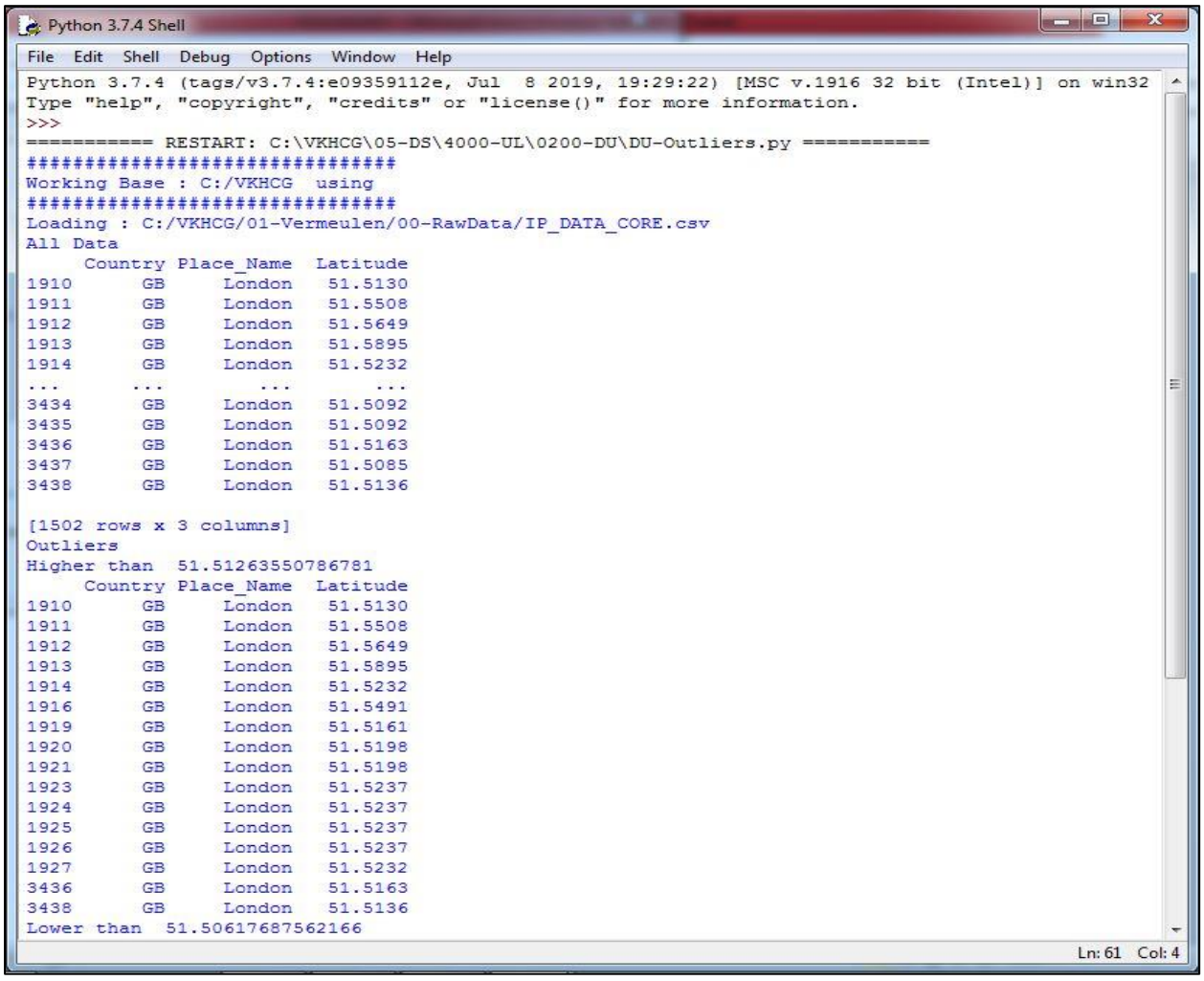
Code:

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

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

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

import pandas as pd

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

InputFileName='IP_DATA_CORE.csv'

OutputFileName='Retrieve_Router_Location.csv'

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

```
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ',)
print('##################################')
#################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data')
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
```

```
OutliersLower=AllData[AllData.Latitude<LowerBound]

print(OutliersLower)

print('Not Outliers')

OutliersNot=AllData[(AllData.Latitude>=LowerBound) &
(AllData.Latitude<=UpperBound)]

print(OutliersNot)

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

Output:



Audit

E. Logging

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

Code:

```
import sys
import os
import logging
import uuid
import shutil
import time
#################################################################
Base='C:/VKHCG'
#################################################################
sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04-Clark']
sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-
Organise','06-Report']
sLevels=['debug','info','warning','error']
for sCompany in sCompanies:
   sFileDir=Base + '/' + sCompany
   if not os.path.exists(sFileDir):
      os.makedirs(sFileDir)
```

```
    for sLayer in sLayers:
        log = logging.getLogger()  # root logger
        for hdlr in log.handlers[:]:  # remove all old handlers
            log.removeHandler(hdlr)
        ################################################################
        sFileDir=Base + '/' + sCompany + '/' + sLayer + '/Logging'
        if os.path.exists(sFileDir):
            shutil.rmtree(sFileDir)
        time.sleep(2)
        if not os.path.exists(sFileDir):
            os.makedirs(sFileDir)
        skey=str(uuid.uuid4())
        sLogFile=Base + '/' + sCompany + '/' + sLayer +
'/Logging/Logging_'+skey+'.log'
        print('Set up:',sLogFile)
        # set up logging to file - see previous section for more details
        logging.basicConfig(level=logging.DEBUG,
                            format='%(asctime)s %(name)-12s %(levelname)-8s
%(message)s',
                            datefmt='%m-%d %H:%M',
                            filename=sLogFile,
                            filemode='w')
        # define a Handler which writes INFO messages or higher to the sys.stderr
        console = logging.StreamHandler()
```

```
    console.setLevel(logging.INFO)
    # set a format which is simpler for console use
    formatter = logging.Formatter('%(name)-12s: %(levelname)-8s
%(message)s')
    # tell the handler to use this format
    console.setFormatter(formatter)
    # add the handler to the root logger
    logging.getLogger('').addHandler(console)
    # Now, we can log to the root logger, or any other logger. First the root...
    logging.info('Practical Data Science is fun!.')
    for sLevel in sLevels:
      sApp='Apllication-'+ sCompany + '-' + sLayer + '-' + sLevel
      logger = logging.getLogger(sApp)
      if sLevel == 'debug':
        logger.debug('Practical Data Science logged a debugging message.')
      if sLevel == 'info':
        logger.info('Practical Data Science logged information message.')
      if sLevel == 'warning':
        logger.warning('Practical Data Science logged a warning message.')
      if sLevel == 'error':
        logger.error('Practical Data Science logged an error message.')
##################################################################
```

Output:


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Apllication-03-Hillman-04-Transform-info: INFO Practical Data Science logged information message.
Apllication-03-Hillman-04-Transform-warning: WARNING Practical Data Science logged a warning message.
Apllication-03-Hillman-04-Transform-error: ERROR Practical Data Science logged an error message.
Set up: C:/VKHCG/03-Hillman/05-Organise/Logging/Logging_28c2d1d3-52b8-4773-a030-076def7b60e7.log
root : INFO Practical Data Science is fun!.
Apllication-03-Hillman-05-Organise-info: INFO Practical Data Science logged information message.
Apllication-03-Hillman-05-Organise-warning: WARNING Practical Data Science logged a warning message.
Apllication-03-Hillman-05-Organise-error: ERROR Practical Data Science logged an error message.
Set up: C:/VKHCG/03-Hillman/06-Report/Logging/Logging_3df0472e-fa4f-4da0-9456-dcf8a615c0d6.log
root : INFO Practical Data Science is fun!.
Apllication-03-Hillman-06-Report-info: INFO Practical Data Science logged information message.
Apllication-03-Hillman-06-Report-warning: WARNING Practical Data Science logged a warning message.
Apllication-03-Hillman-06-Report-error: ERROR Practical Data Science logged an error message.
Set up: C:/VKHCG/04-Clark/01-Retrieve/Logging/Logging_806210b1-51b0-42fb-9625-18d54a85e20e.log
root : INFO Practical Data Science is fun!.
Apllication-04-Clark-01-Retrieve-info: INFO Practical Data Science logged information message.
Apllication-04-Clark-01-Retrieve-warning: WARNING Practical Data Science logged a warning message.
Apllication-04-Clark-01-Retrieve-error: ERROR Practical Data Science logged an error message.
Set up: C:/VKHCG/04-Clark/02-Assess/Logging/Logging_53601e70-0a50-4a3e-a392-3533bd9c3f03.log
root : INFO Practical Data Science is fun!.
Apllication-04-Clark-02-Assess-info: INFO Practical Data Science logged information message.
Apllication-04-Clark-02-Assess-warning: WARNING Practical Data Science logged a warning message.
Apllication-04-Clark-02-Assess-error: ERROR Practical Data Science logged an error message.
Set up: C:/VKHCG/04-Clark/03-Process/Logging/Logging_1ba52634-d2a5-46eb-a769-f7fd9a5cbcc4.log
root : INFO Practical Data Science is fun!.
Apllication-04-Clark-03-Process-info: INFO Practical Data Science logged information message.
Apllication-04-Clark-03-Process-warning: WARNING Practical Data Science logged a warning message.
Apllication-04-Clark-03-Process-error: ERROR Practical Data Science logged an error message.
Set up: C:/VKHCG/04-Clark/04-Transform/Logging/Logging_b22c5fc9-19c3-4ea1-bd64-ec2cdc93eb46.log
root : INFO Practical Data Science is fun!.
Apllication-04-Clark-04-Transform-info: INFO Practical Data Science logged information message.
Apllication-04-Clark-04-Transform-warning: WARNING Practical Data Science logged a warning message.
Apllication-04-Clark-04-Transform-error: ERROR Practical Data Science logged an error message.
Set up: C:/VKHCG/04-Clark/05-Organise/Logging/Logging_754fb8c4-611d-4e67-b23e-6c27d8e84a82.log
root : INFO Practical Data Science is fun!.
Apllication-04-Clark-05-Organise-info: INFO Practical Data Science logged information message.
Apllication-04-Clark-05-Organise-warning: WARNING Practical Data Science logged a warning message.
Apllication-04-Clark-05-Organise-error: ERROR Practical Data Science logged an error message.
Set up: C:/VKHCG/04-Clark/06-Report/Logging/Logging_389f3c92-e797-4fcd-b7ac-9515b26d4a0d.log
root : INFO Practical Data Science is fun!.
Apllication-04-Clark-06-Report-info: INFO Practical Data Science logged information message.
Apllication-04-Clark-06-Report-warning: WARNING Practical Data Science logged a warning message.
Apllication-04-Clark-06-Report-error: ERROR Practical Data Science logged an error message.
>>>
Ln: 123 Col: 97

Practical 4

Retrieve Superstep

A. Program to retrieve different attributes of data.

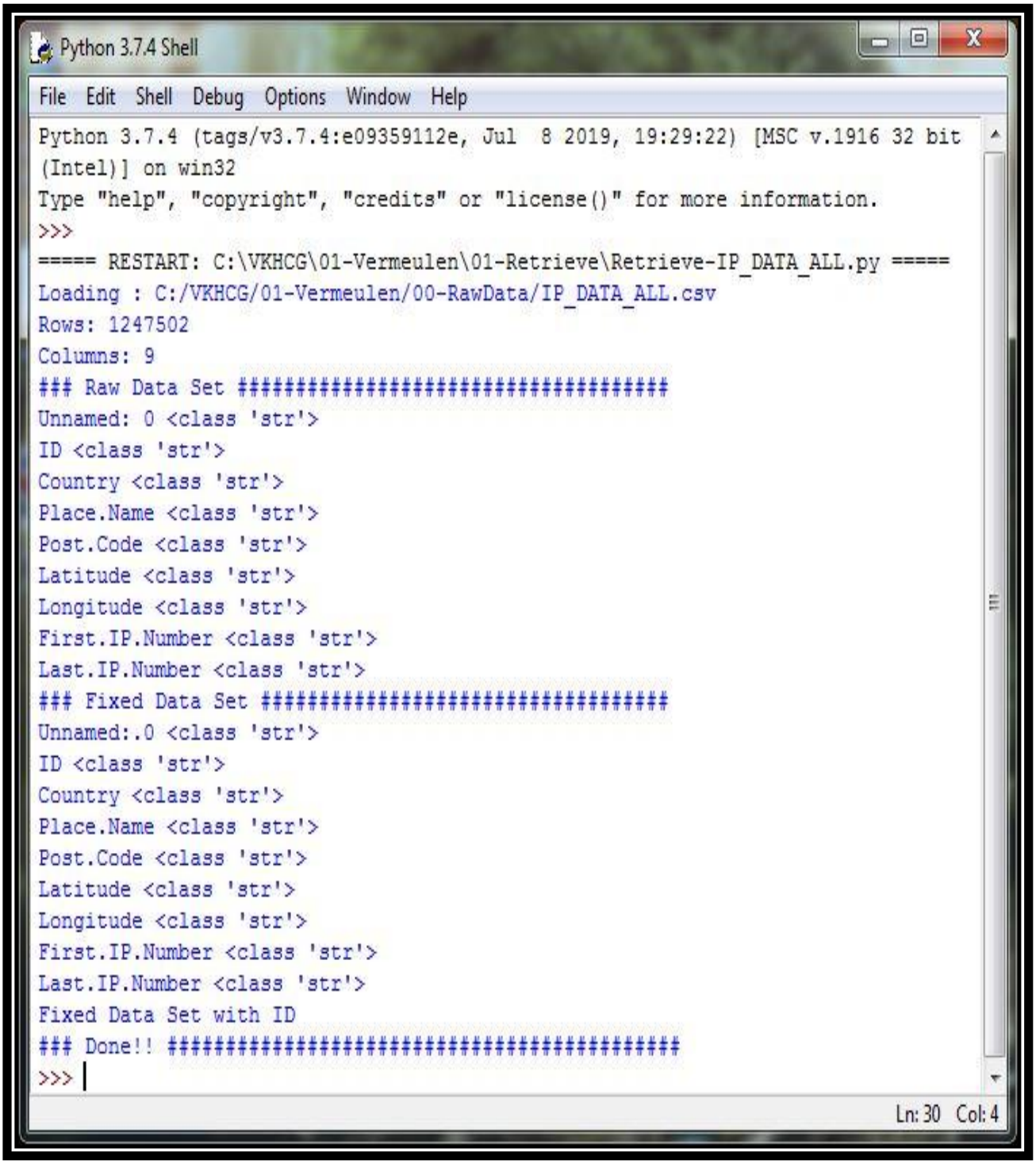
Code:

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


print('Rows:', IP_DATA_ALL.shape[0])
```

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

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

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

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

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

Output:



Practical 5

Assess Superstep

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

Missing Values in Pandas:
i. Drop the Columns Where All Elements Are Missing Values

Code:

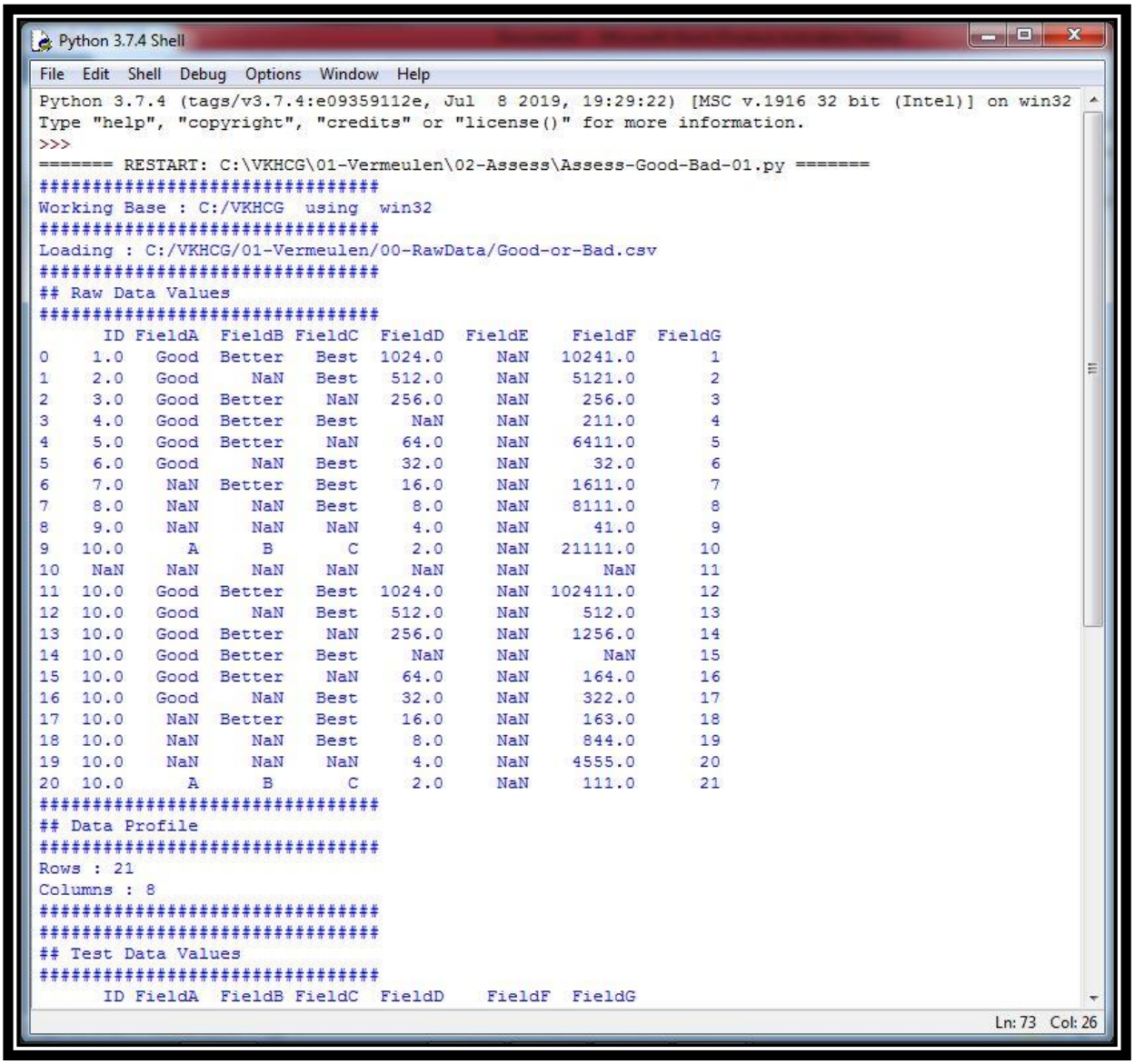
```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
#####################################################################
if sys.platform == 'linux':
   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:



```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-Good-Bad-01.py =======
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
################################
## Raw Data Values
################################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
################################
## Data Profile
################################
Rows : 21
Columns : 8
################################
################################
## Test Data Values
################################
      ID FieldA  FieldB FieldC  FieldD    FieldF  FieldG
                                                  Ln: 73  Col: 26
```

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

Code:

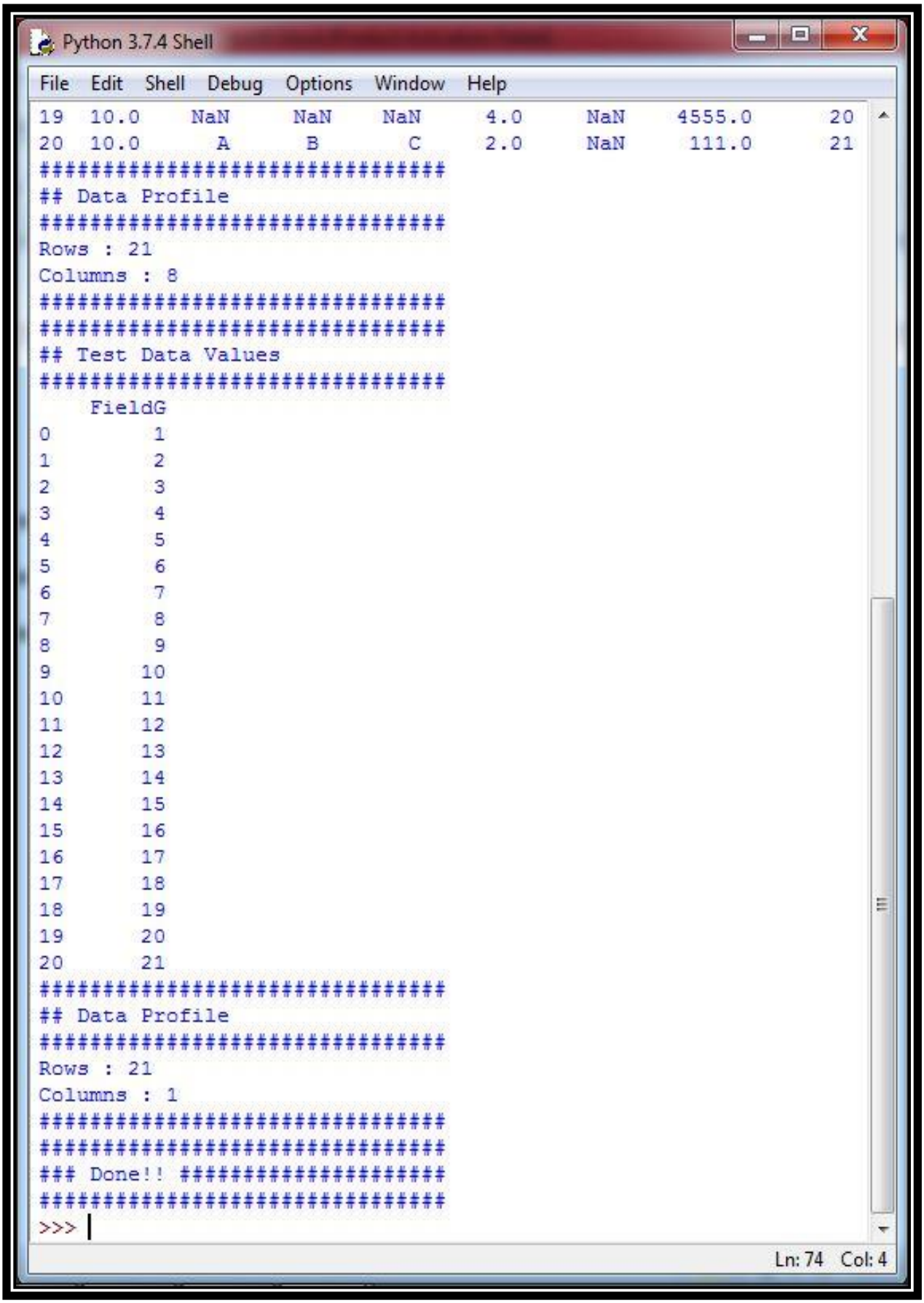
```
#########################################################################

# -*- 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:


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
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!!
##################################
>>>
Ln: 74 Col: 4

Practical 6

Processing Data

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

Code:

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
from datetime import datetime
from datetime import timedelta
from pytz import timezone, all_timezones
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid


pd.options.mode.chained_assignment = None
#####################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
```

```
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
##########################################################################
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
##########################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
##########################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
##########################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
##########################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
##########################################################################
base = datetime(2018,1,1,0,0,0)
```

```
numUnits=10*365*24
#####################################################################
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)]
t=0
for i in date_list:
   now_utc=i.replace(tzinfo=timezone('UTC'))
   sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S")
   print(sDateTime)
   sDateTimeKey=sDateTime.replace(' ','-').replace(':','-')
   t+=1
   IDNumber=str(uuid.uuid4())
   TimeLine=[('ZoneBaseKey', ['UTC']),
         ('IDNumber', [IDNumber]),
         ('nDateTimeValue', [now_utc]),
         ('DateTimeValue', [sDateTime]),
         ('DateTimeKey', [sDateTimeKey])]
   if t==1:
     TimeFrame = pd.DataFrame.from_items(TimeLine)
   else:
      TimeRow = pd.DataFrame.from_items(TimeLine)
      TimeFrame = TimeFrame.append(TimeRow)
#####################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeVal
ue']]
```

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

```
#####################################################################
#
   ###################################################################
   sTable = 'Satellite-Time-'+sZone
   print('Storing :',sDatabaseName,' Table:',sTable)
   TimeZoneFrameIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
#
print('#################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#################')
#####################################################################
#
print('### Done!! ################################################')
#####################################################################
#
```

Output:


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
2017-12-30 18:00:00
2017-12-30 17:00:00
2017-12-30 16:00:00
2017-12-30 15:00:00
2017-12-30 14:00:00
2017-12-30 13:00:00
2017-12-30 12:00:00
2017-12-30 11:00:00
2017-12-30 10:00:00
2017-12-30 09:00:00
2017-12-30 08:00:00
2017-12-30 07:00:00
2017-12-30 06:00:00
2017-12-30 05:00:00
2017-12-30 04:00:00
2017-12-30 03:00:00
2017-12-30 02:00:00
2017-12-30 01:00:00
2017-12-30 00:00:00
2017-12-29 23:00:00
2017-12-29 22:00:00
2017-12-29 21:00:00
2017-12-29 20:00:00
2017-12-29 19:00:00
2017-12-29 18:00:00
2017-12-29 17:00:00
2017-12-29 16:00:00
2017-12-29 15:00:00
2017-12-29 14:00:00
2017-12-29 13:00:00
2017-12-29 12:00:00
2017-12-29 11:00:00
2017-12-29 10:00:00
2017-12-29 09:00:00
2017-12-29 08:00:00
2017-12-29 07:00:00
2017-12-29 06:00:00
2017-12-29 05:00:00
2017-12-29 04:00:00
2017-12-29 03:00:00Traceback (most recent call last):
File "C:\VKHCG\01-Vermeulen\03-Process\Process_Time.py", line 50, in <module>
print(sDateTime)
KeyboardInterrupt
Ln: 91 Col: 4

Practical 7

Transform Superstep

A: To illustrate the consolidation process, the example show a person being borne.

Open a new file in the Python editor and save it as Transform-Gunnarsson_is_Born.py in directory

Transform-Gunnarsson_is_Born.py

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)
#########################################################################
```

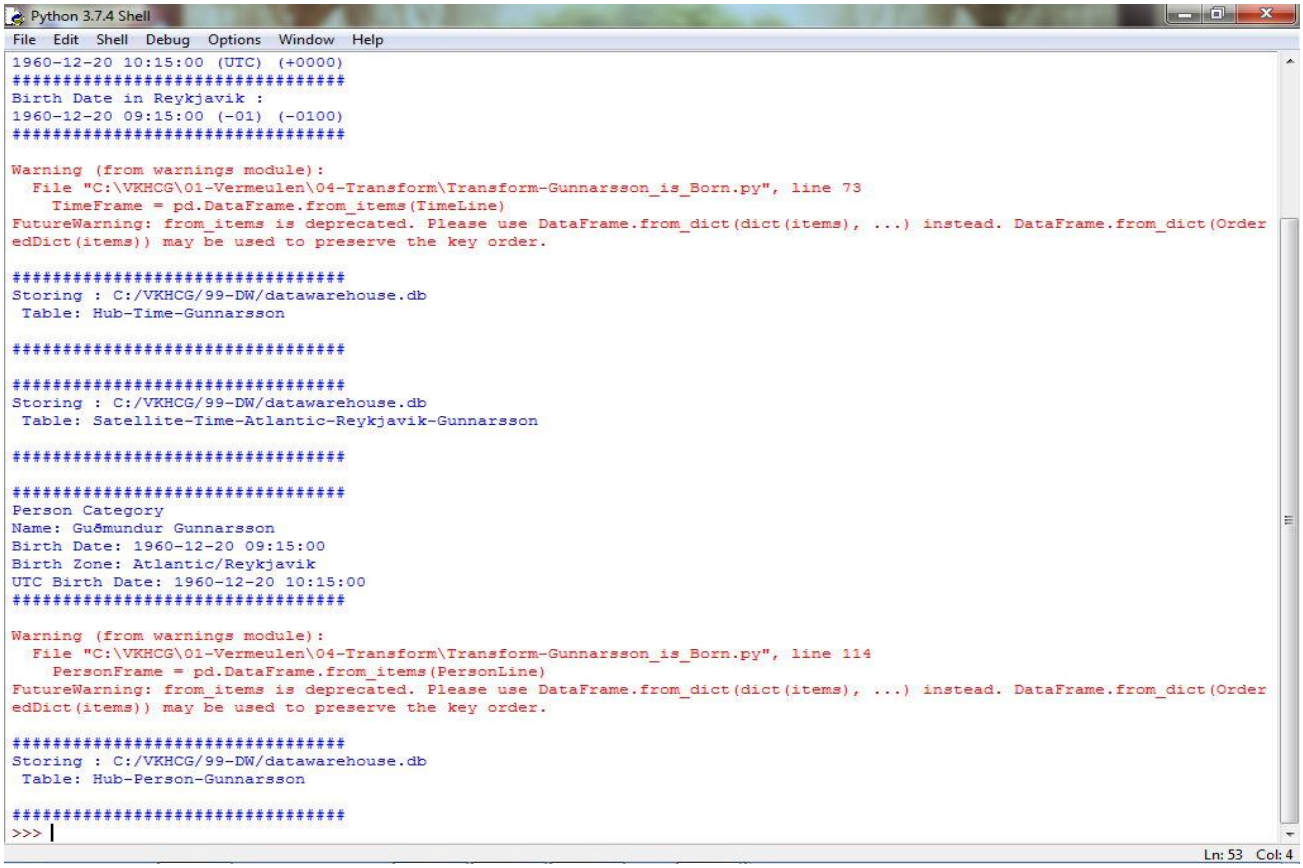
```
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
   os.makedirs(sDataWarehouseDir)
######################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
######################################################################
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','DateTimeVal
ue']]
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:



B: You must build three items: dimension Person, dimension Time, and factPersonBornAtTime.

Open your Python editor and create a file named Transform-Gunnarsson-Sun-Model.py in directory

Transform-Gunnarsson-Sun-Model.py

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)
#####################################################################
```

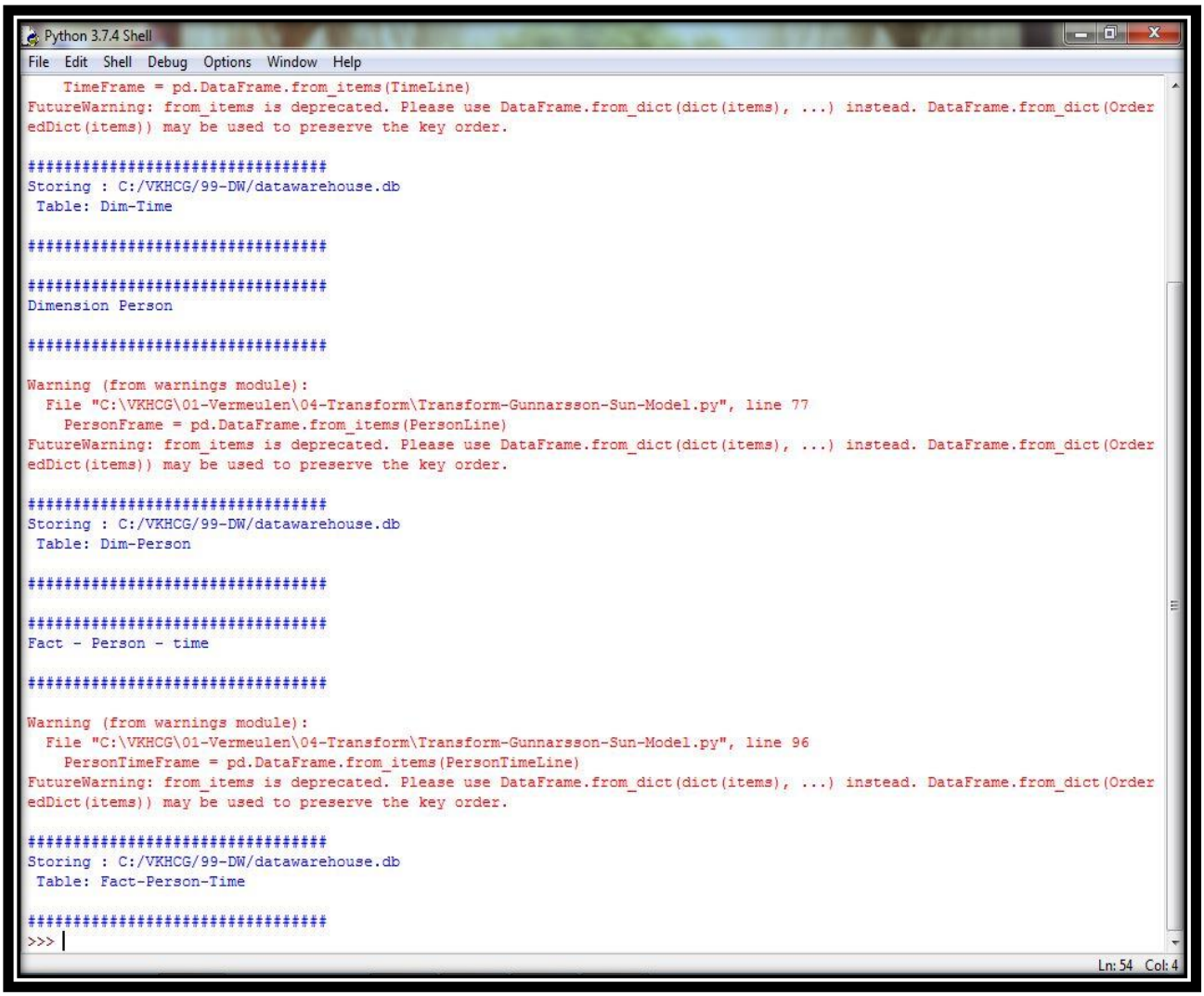
```
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir):
   os.makedirs(sDataWarehousetDir)
#####################################################################
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
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:



Practical 8

Organize Superstep

Horizontal Style

Code:

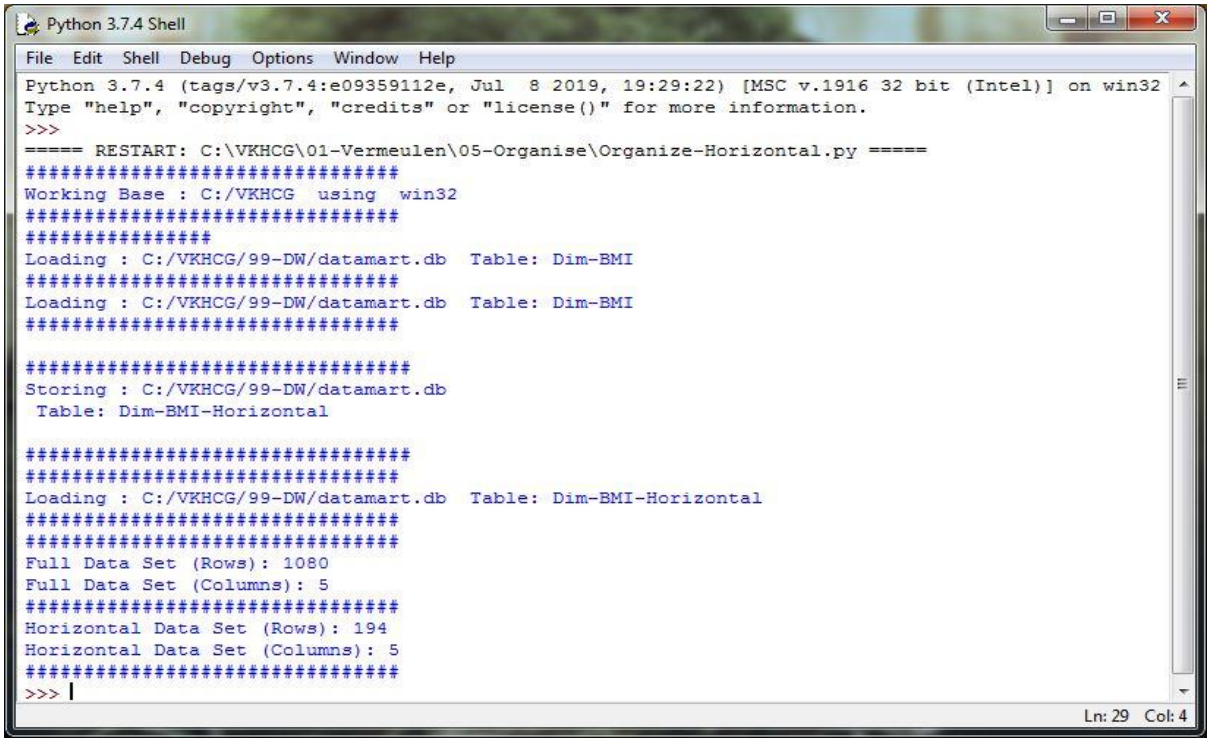
```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
```

```
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
   os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####################################################################
print('#################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#################################')
sSQL="SELECT PersonID,\
     Height,\
```

```
    Weight,\
    bmi,\
    Indicator\
 FROM [Dim-BMI]\
 WHERE \
 Height > 1.5 \
 and Indicator = 1\
 ORDER BY  \
    Height,\
    Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####################################################################
sTable = 'Dim-BMI-Horizontal'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('###################################')
sTable = 'Dim-BMI-Horizontal'
```

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

Output:



The horizontal-style slicing selects the 194 subset of rows from the 1080 rows while preserving the columns.

Vertical Style

Code:

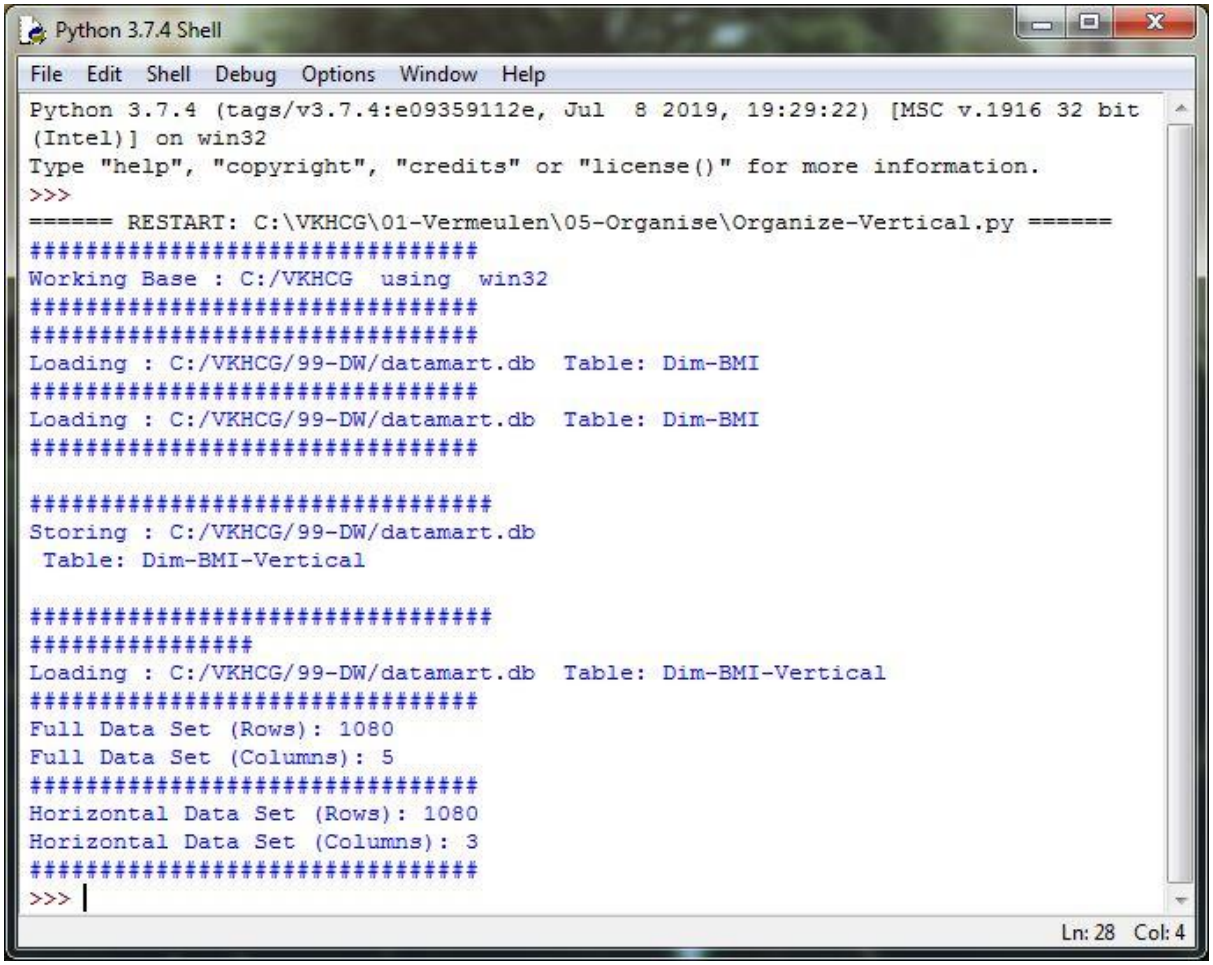
```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
if 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:



The vertical-style slicing selects 3 of 5 from the population, while preserving the rows [1080].

Island Style

Code:

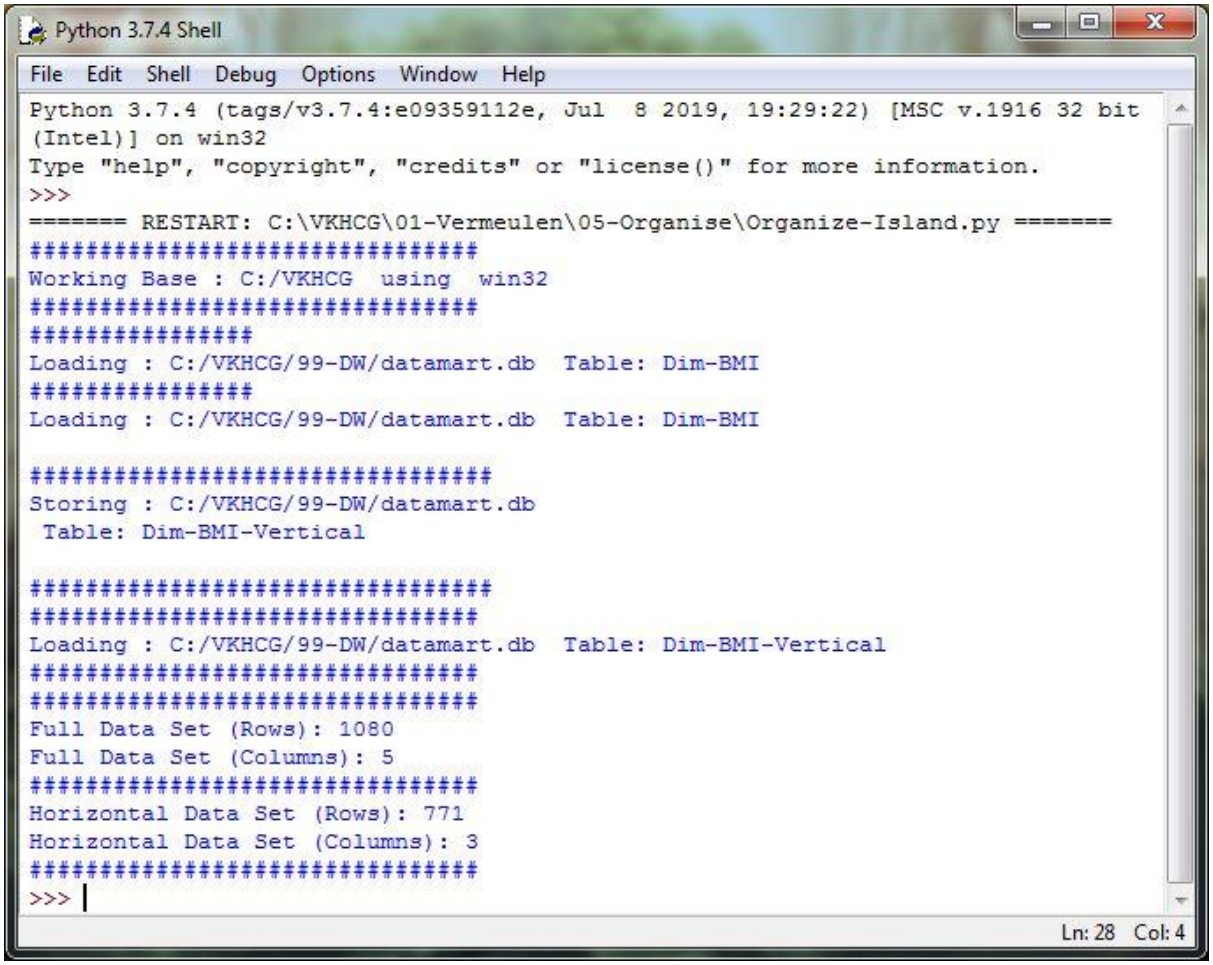
```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
if 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:



This generates a subset of 771 rows out of 1080 rows and 3 columns out of 5.

Secure Vault Style

Code:

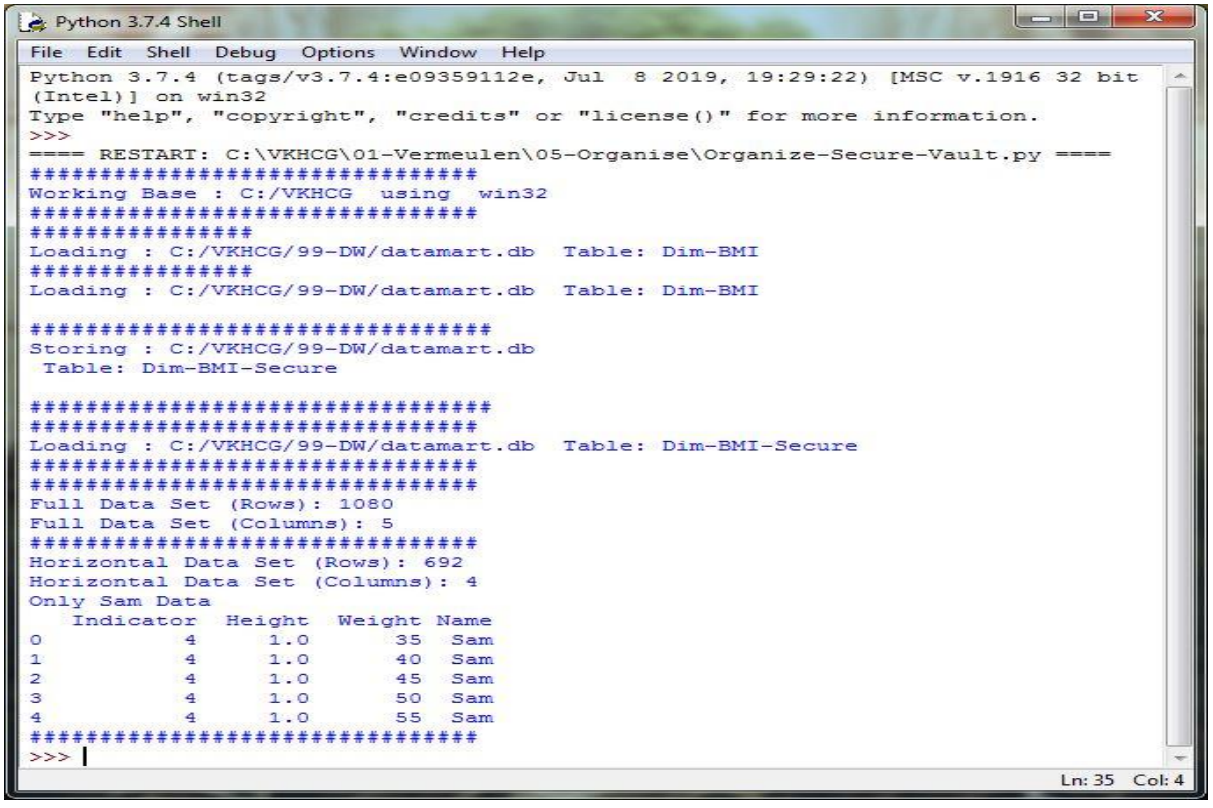
```
#########################################################################
# -*- coding: utf-8 -*-
#########################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
#########################################################################
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())
print('###################################')
#####################################################################
```

Output:



Practical 9

Report Superstep(**Generating Data**)

Vermeulen PLC

Raport-Network-Routing-Customer.py

Code:

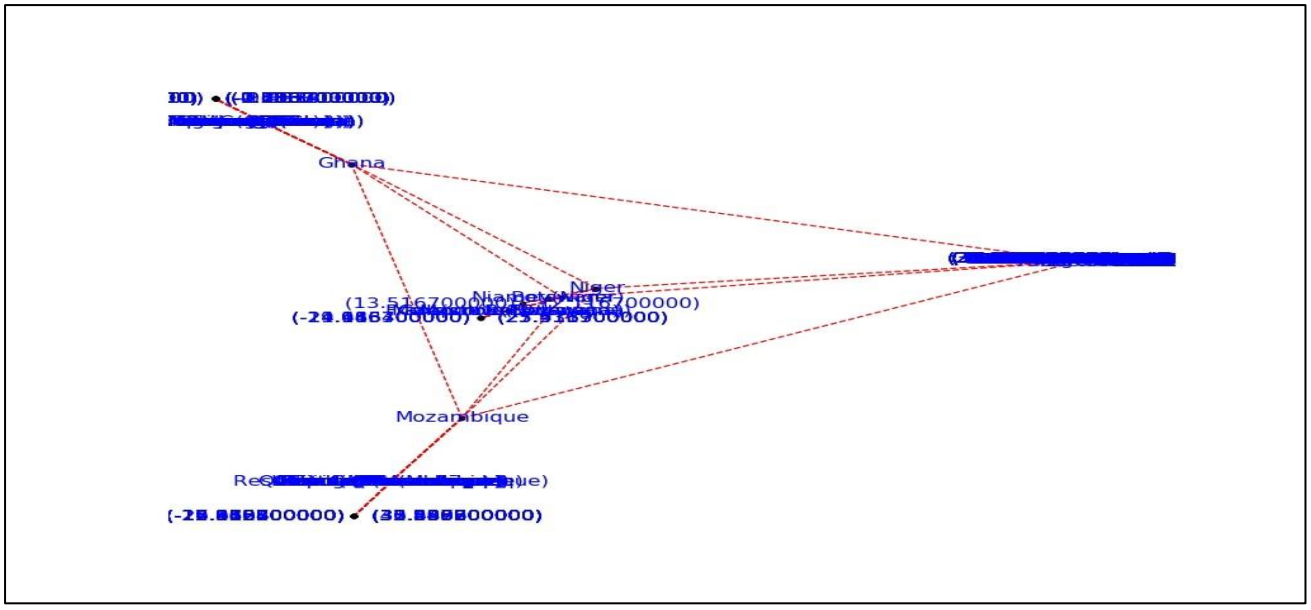
```
#################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
#################################################################
pd.options.mode.chained_assignment = None
#################################################################
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:




```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\VKHCG\01-Vermeulen\06-Report\Raport-Network-Routing-Customer.py
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routin
g-Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitud
e'
 '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-Routin
g-Customer.gml
################################
################################
Storing Graph Image: C:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Net
work-Routing-Customer.png
################################
################################
### Done!! ####################
################################
>>>
                                                                  Ln: 34  Col: 4
```

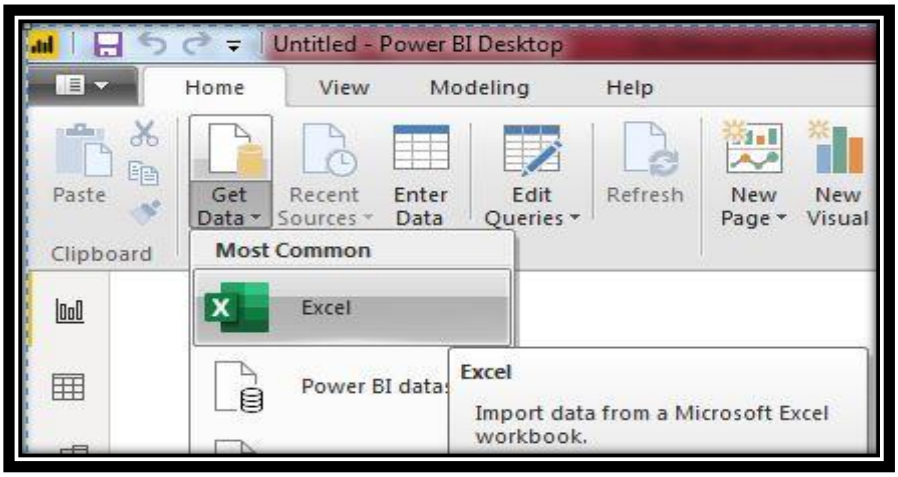
Practical 10
Data Visualization with Power BI

Case Study : Sales Data

Task 1:Importing Data from excel

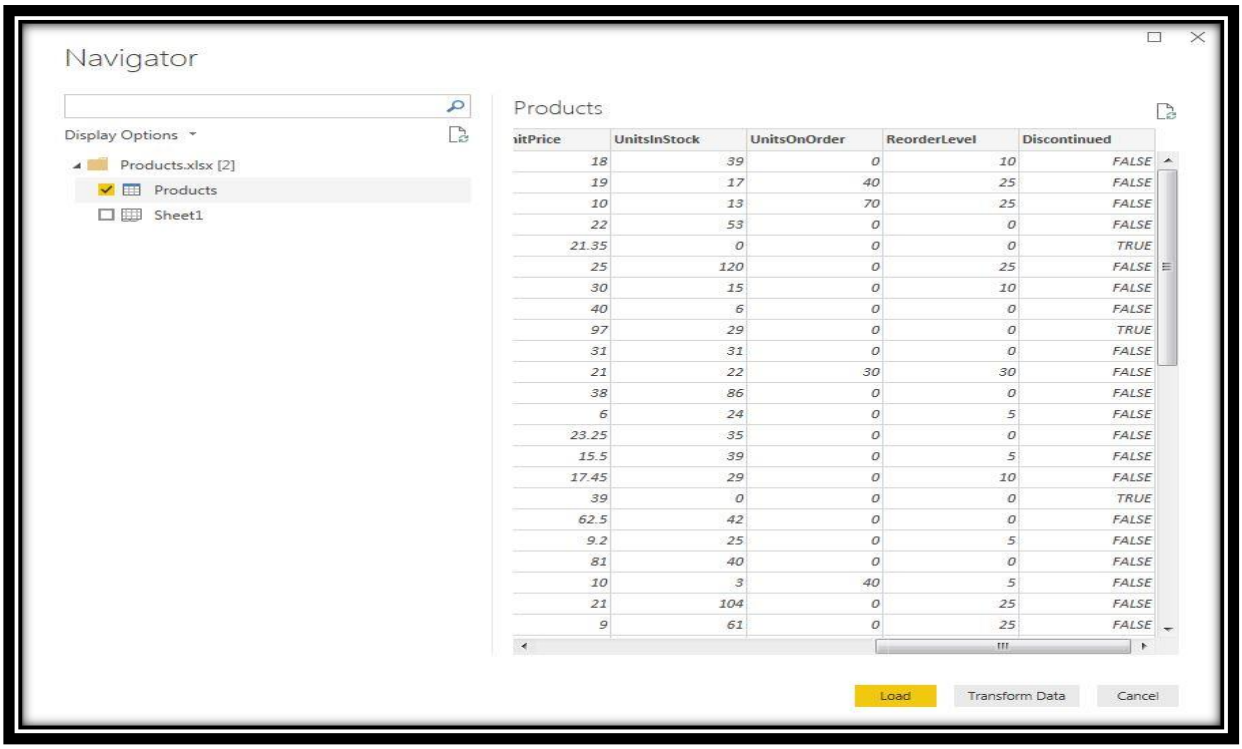
Step 1: Connect to an excel workbook.

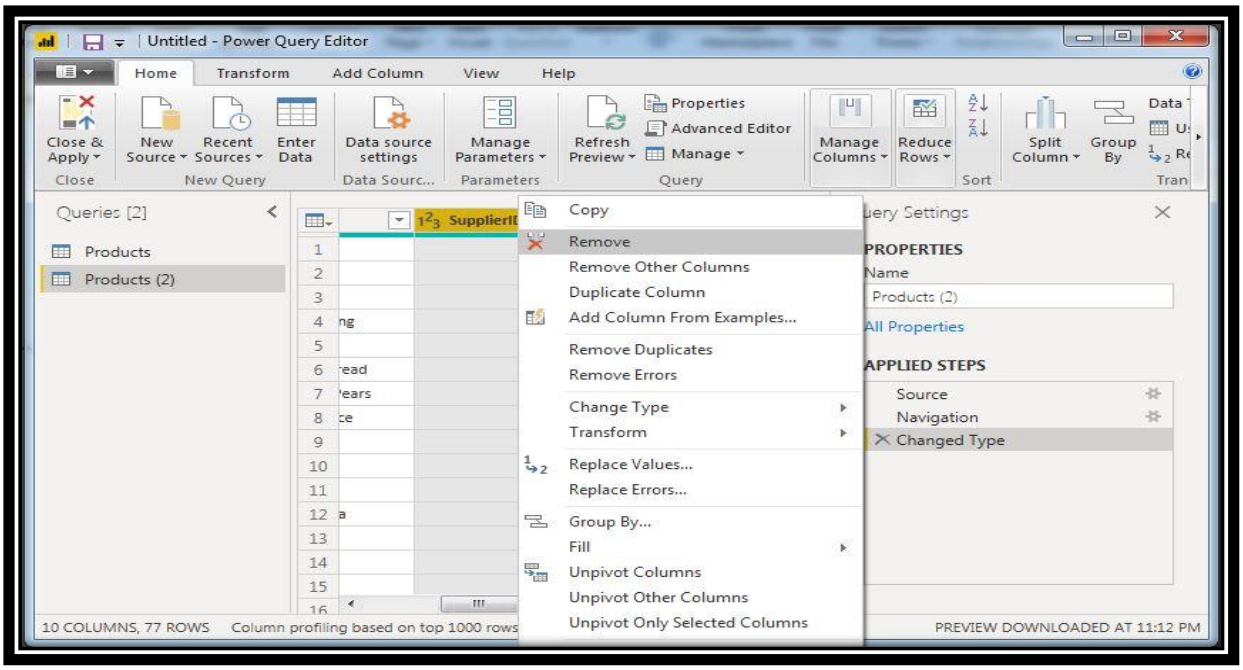
1. Launch Power BI Desktop
2. From the Home ribbon, select Get Data. Excel is one of the Most Common data connections, so you can select if directly from the Get Data menu.



3. If you select the Get Data button directly, you can also select File>Excel and select connect.
4. In the Open File dialog box, select the Product.xlsx file

Step 2: We need to remove other columns except ProductID, ProductName, QuantityPerUnit and UnitInStock

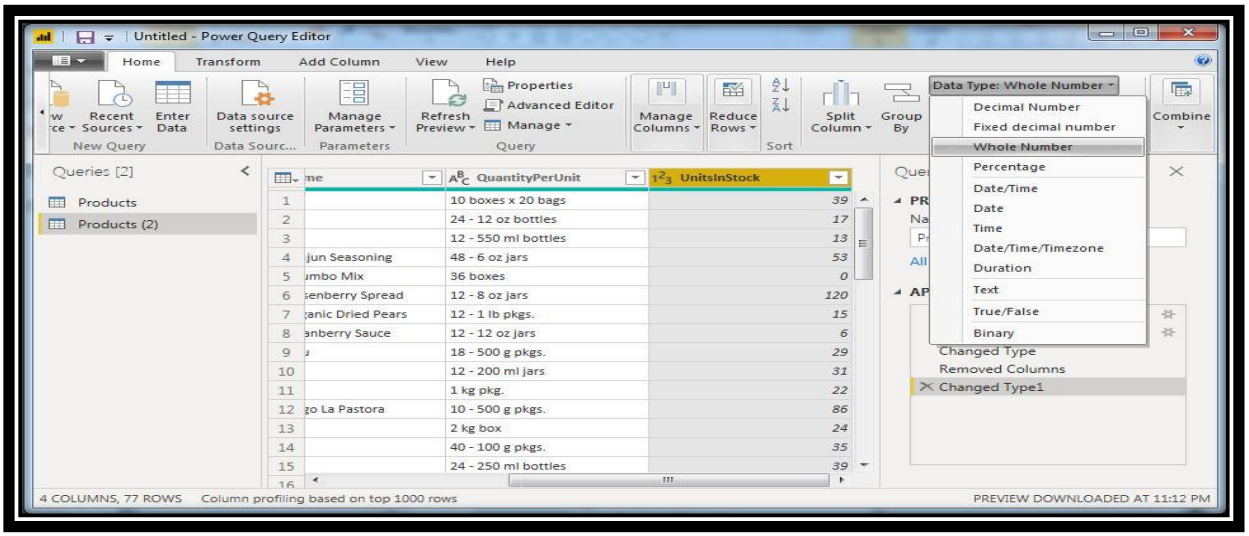




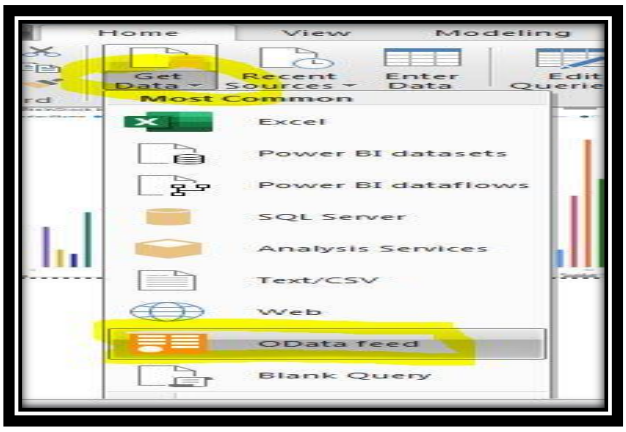
Step 3: Change the data type of the UnitsInStock column

For the Excel workbook, products in stock will always be a whole number, so in this step you confirm the UnitsInStock column's datatype is Whole Number.

1. Select the UnitsInStock column.
2. Select the Data Type drop-down button in the Home ribbon.
3. If not already a Whole Number, select Whole Number for data type from the drop down (Data Type: button also displays the data type for the current selection).



Task 2: Import order data from an OData feed

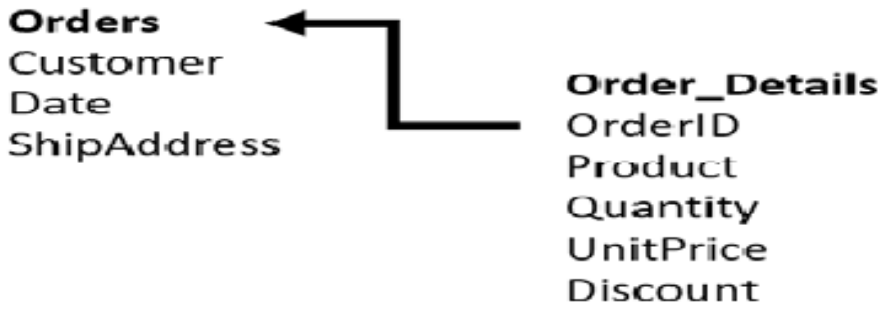


You import data into Power BI Desktop from the sample Northwind OData feed at the following
URL, which you can copy (and then paste) in the steps below:
(http://services.odata.org/V3/Northwind/Northwind.svc/)

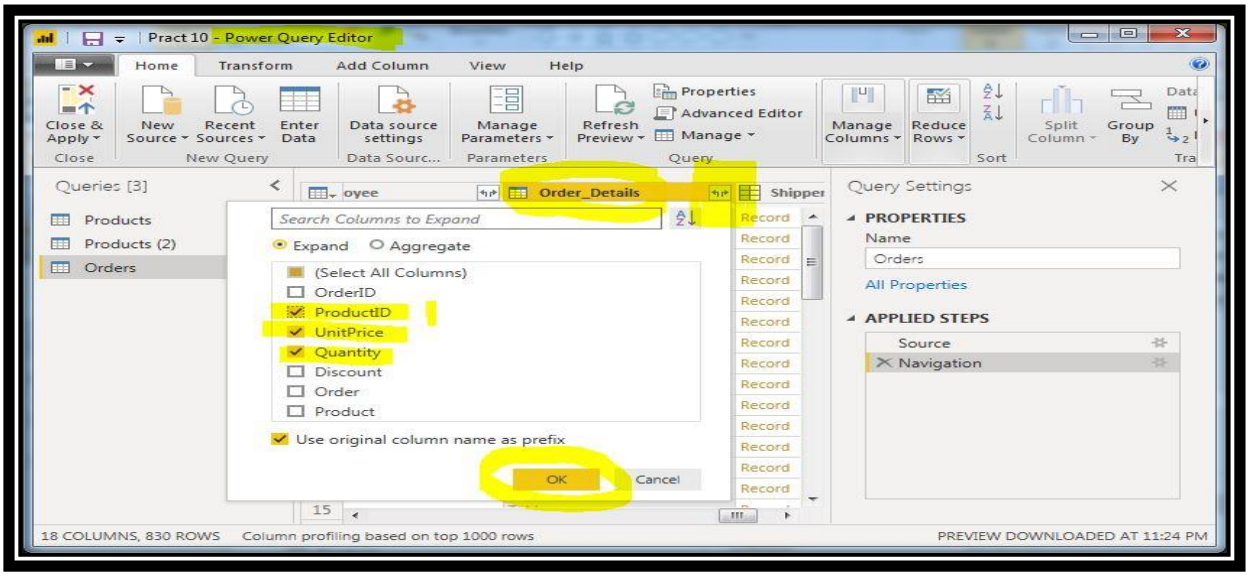
Step 1: Connect to an OData feed

1. From the Home ribbon tab in Query Editor, select Get Data.
2. Browse to the OData Feed data source.
3. In the OData Feed dialog box, paste the URL for the Northwind OData feed.
4. Select OK.

Step 2: Expand the Order_Details table



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



The Expand operation combines columns from a related table into a subject table. When the query runs, rows from the related table (Order_Details) are combined into rows from the subject table (Orders).
After you expand the Order_Details table, three new columns and additional rows are added to the Orders table, one for each row in the nested or related table.

1. In the Query View, scroll to the Order_Details column.
2. In the Order_Details column, select the expand icon ().
3. In the Expand drop-down: a. Select (Select All Columns) to clear all columns.

Select ProductID, UnitPrice, and Quantity.
click OK.

Step 3: Remove other columns to only display columns of interest

In this step you remove all columns except OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and Order_Details.Quantity columns. In the previous task, you used Remove Other Columns. For this task, you remove selected columns.
In the Query View, select all columns by completing a.

a. Click the first column (OrderID).
b. Shift+Click the last column (Shipper).
c. Now that all columns are selected, use Ctrl+Click to unselect the following columns:

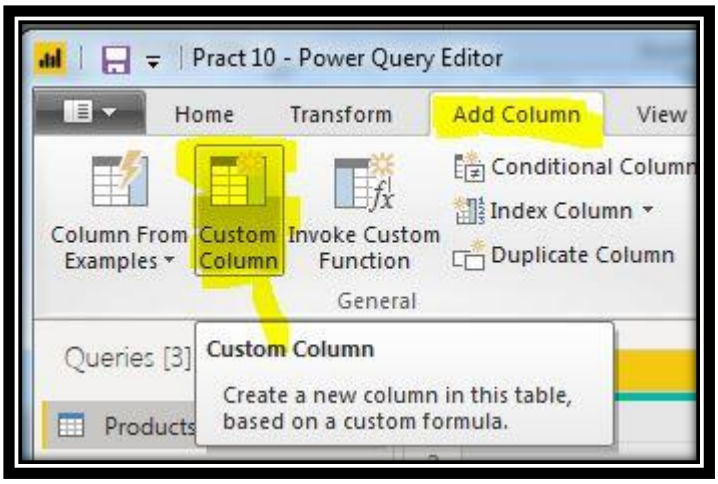
OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and Order_Details.Quantity.

Now that only the columns we want to remove are selected, right-click on any selected column header and click Remove Columns.

Step 4: Calculate the line total for each Order_Details row

Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order_Details row.
Calculate the line total for each Order_Details row:

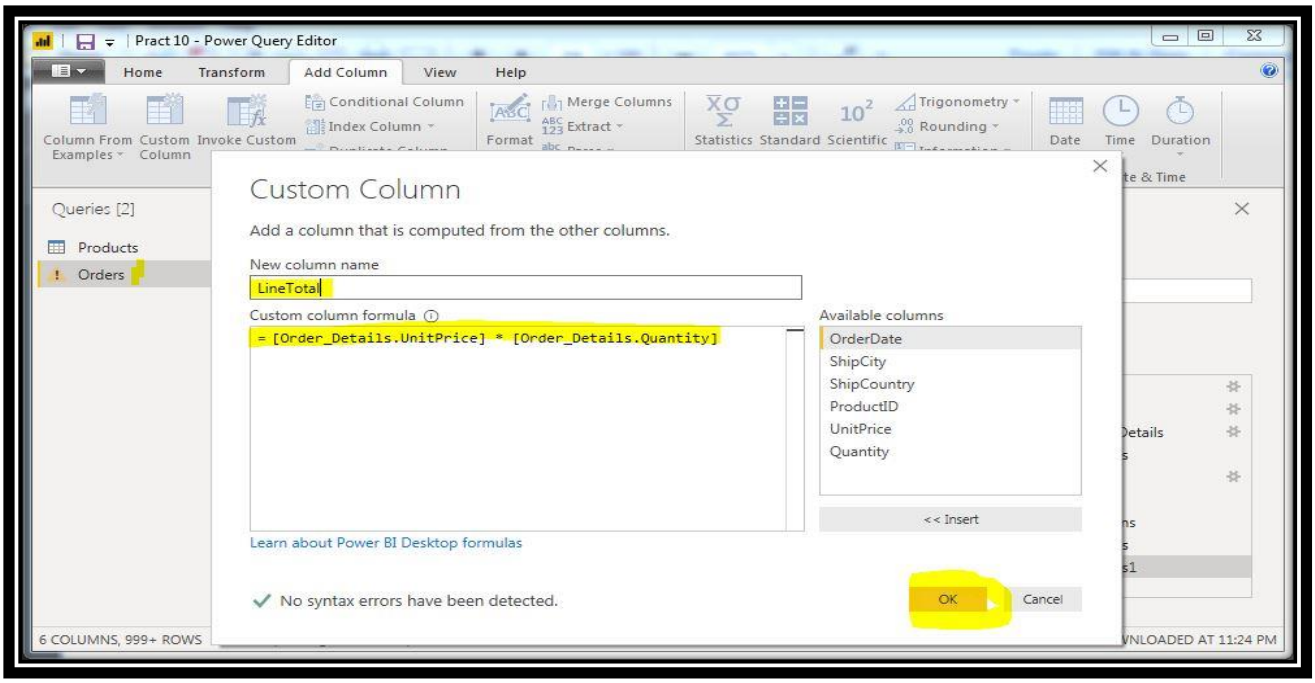
1. In the Add Column ribbon tab, click Add Custom Column.



2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter

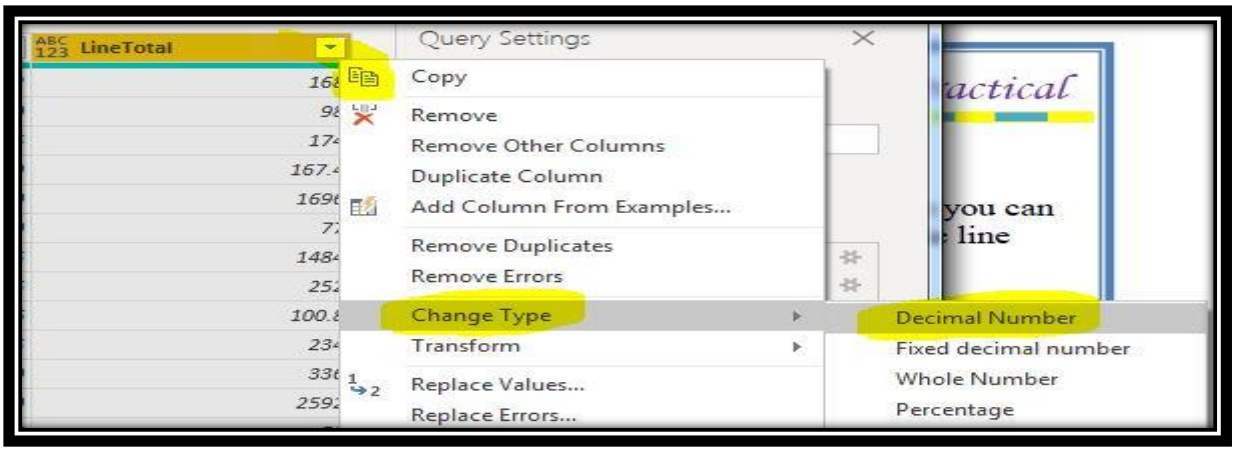
[Order_Details.UnitPrice] * [Order_Details.Quantity].

3. In the New column name textbox, enter LineTotal.



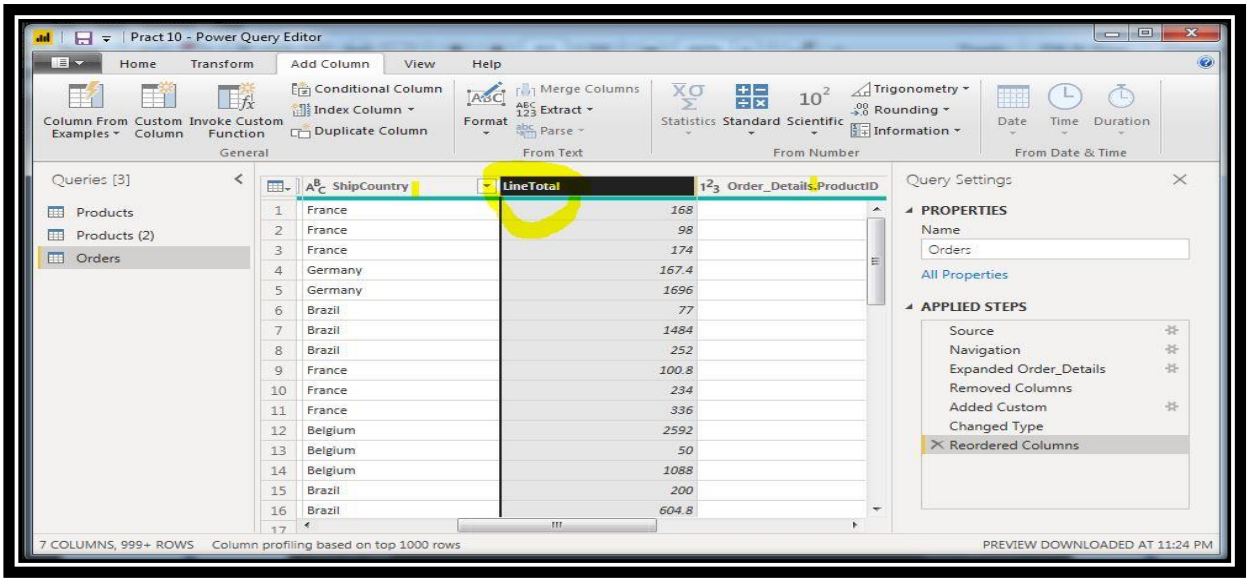
Step 5: Set the datatype of the LineTotal field

1. Right click the LineTotal column.
2. Select Change Type and choose Decimal Number.



Step 6: Rename and reorder columns in the query

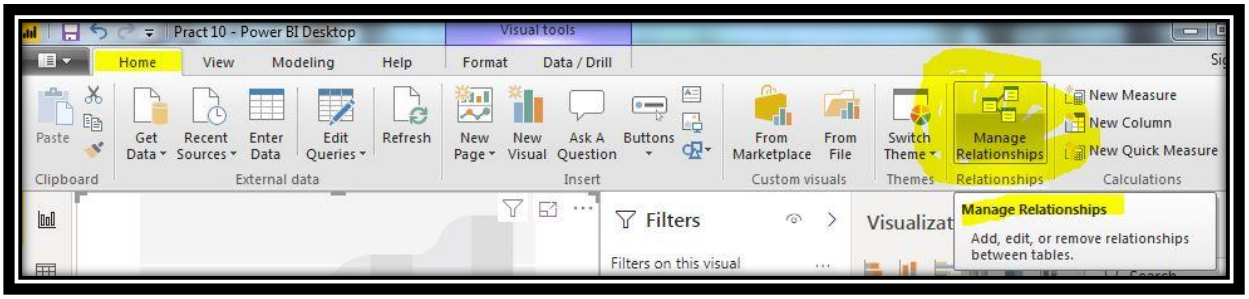
1. In Query Editor, drag the LineTotal column to the left, after ShipCountry.



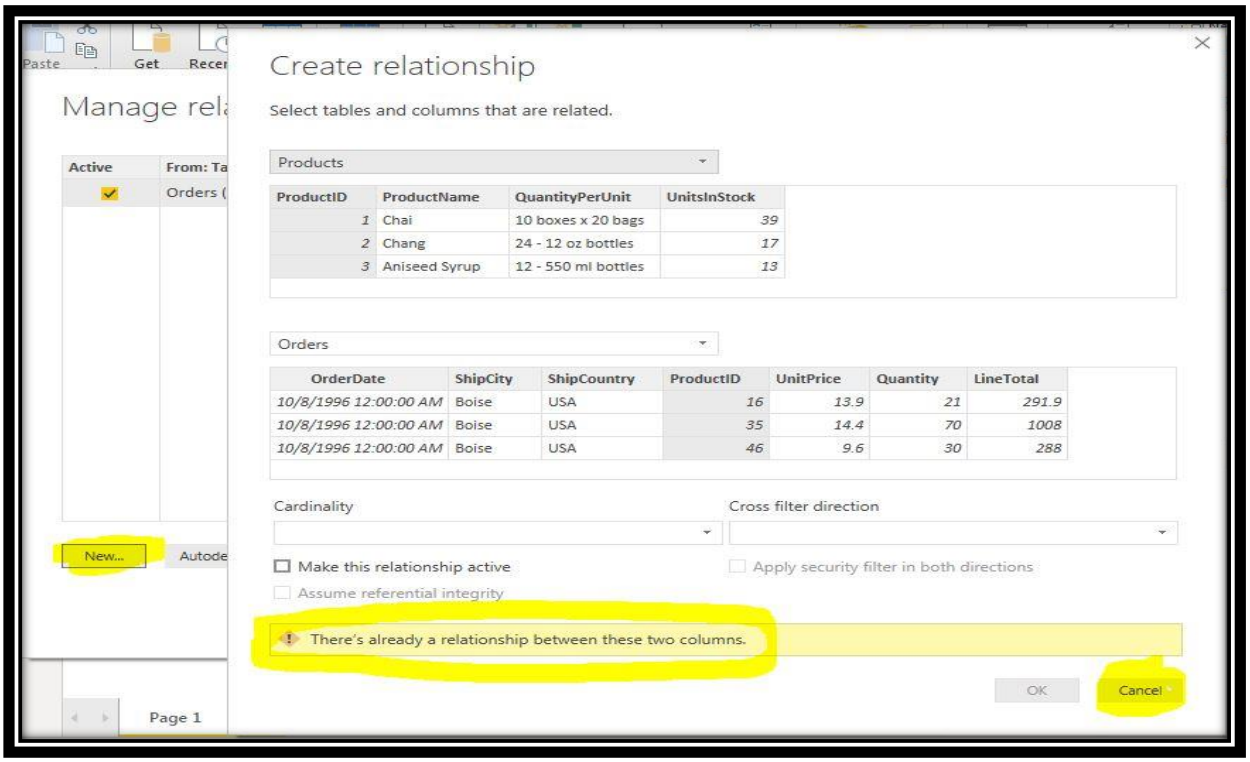
2. 2.Remove the Order_Details.prefix from the Order_Details.ProductID, Order_Details.UnitPrice and Order_Details.Quantity columns, by double-clicking on each column header, and then deleting that text from the column name.

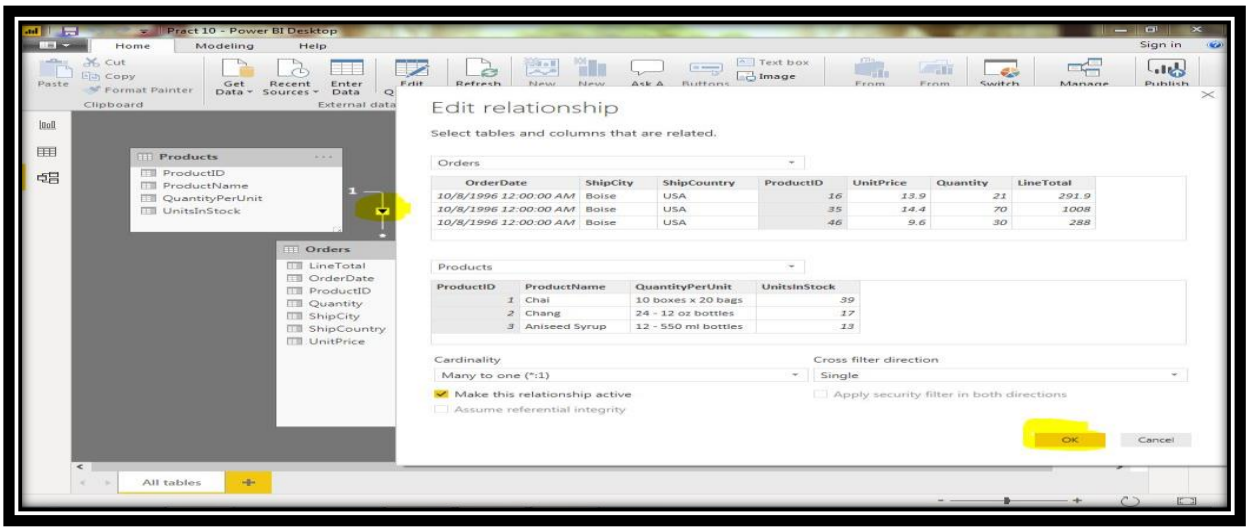
Task 3: Combine the Products and Total Sales queries

1. Power BI Desktop loads the data from the two queries
2. Once the data is loaded, select the Manage Relationships button Home ribbon



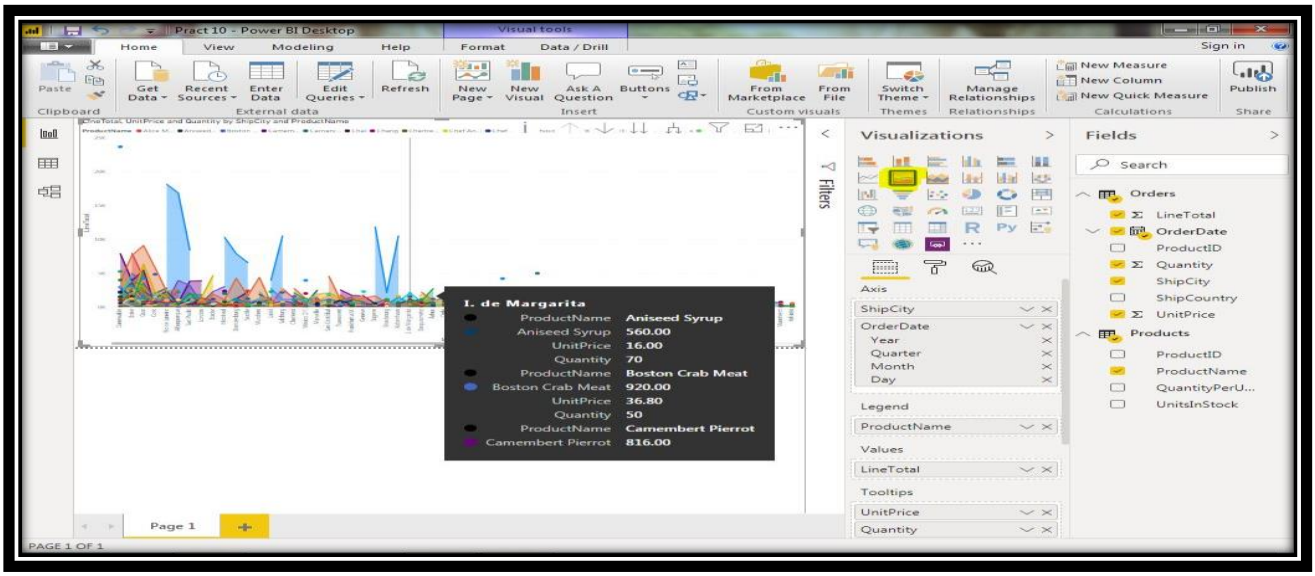
3. Select the New… button
4. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.
5. Select Cancel, and then select Relationship view in Power BI Desktop.



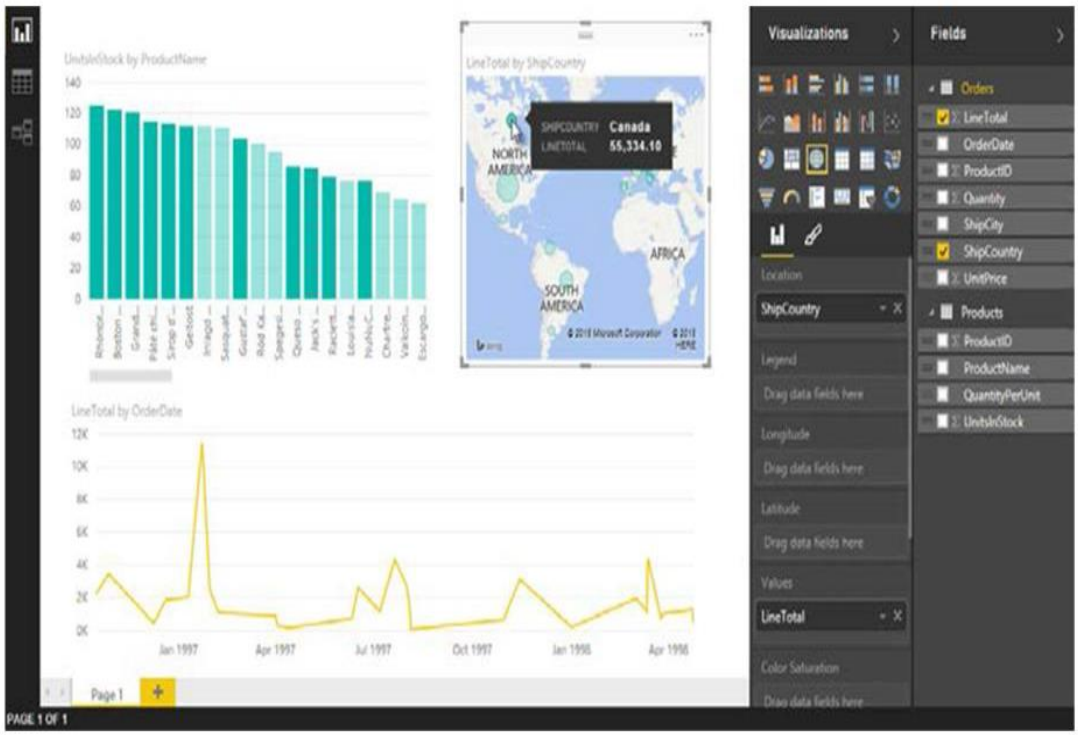


Task 4: Build visuals using your data

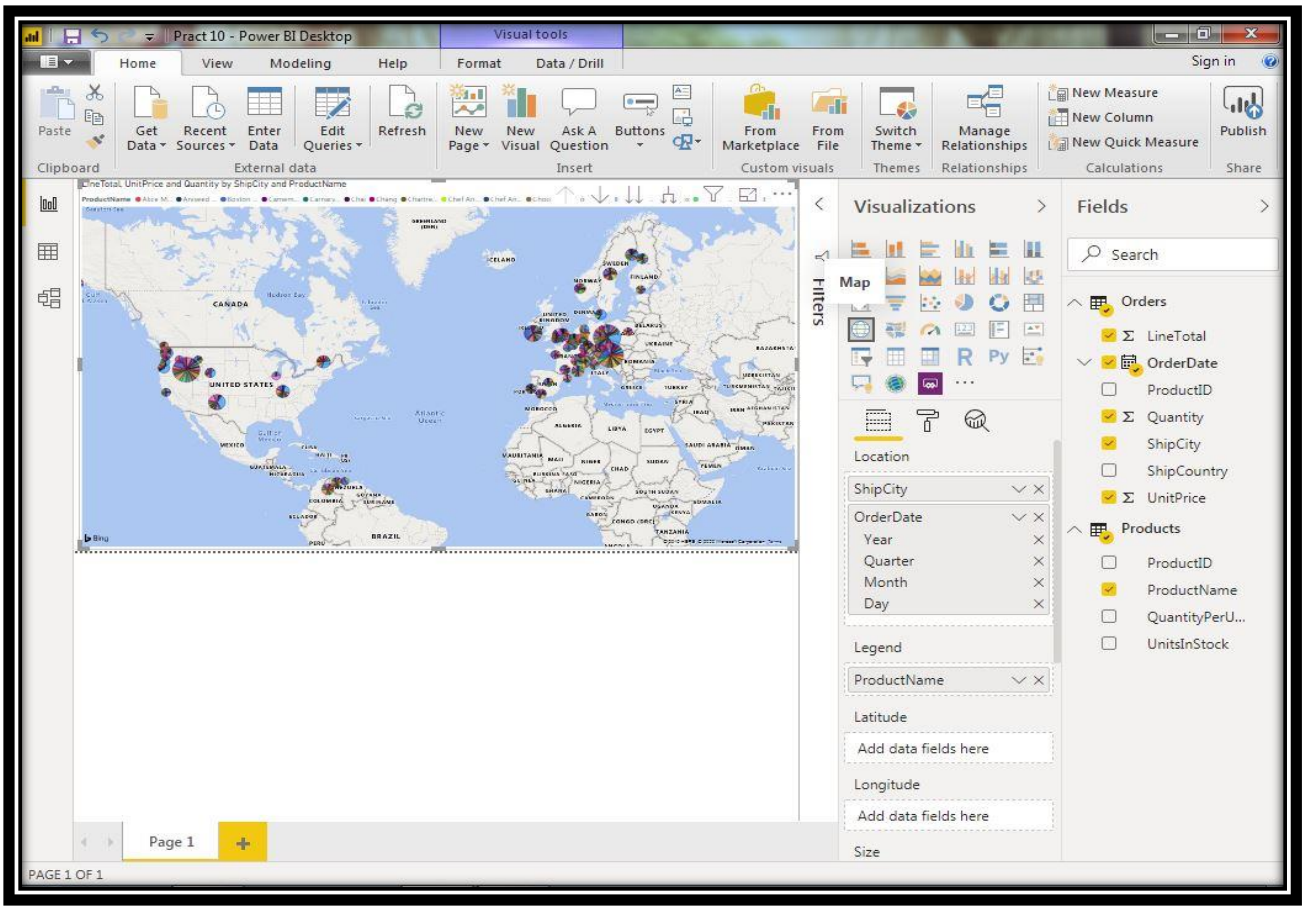
Step 1: Create charts showing Units in Stock by Product and Total Sales by Year

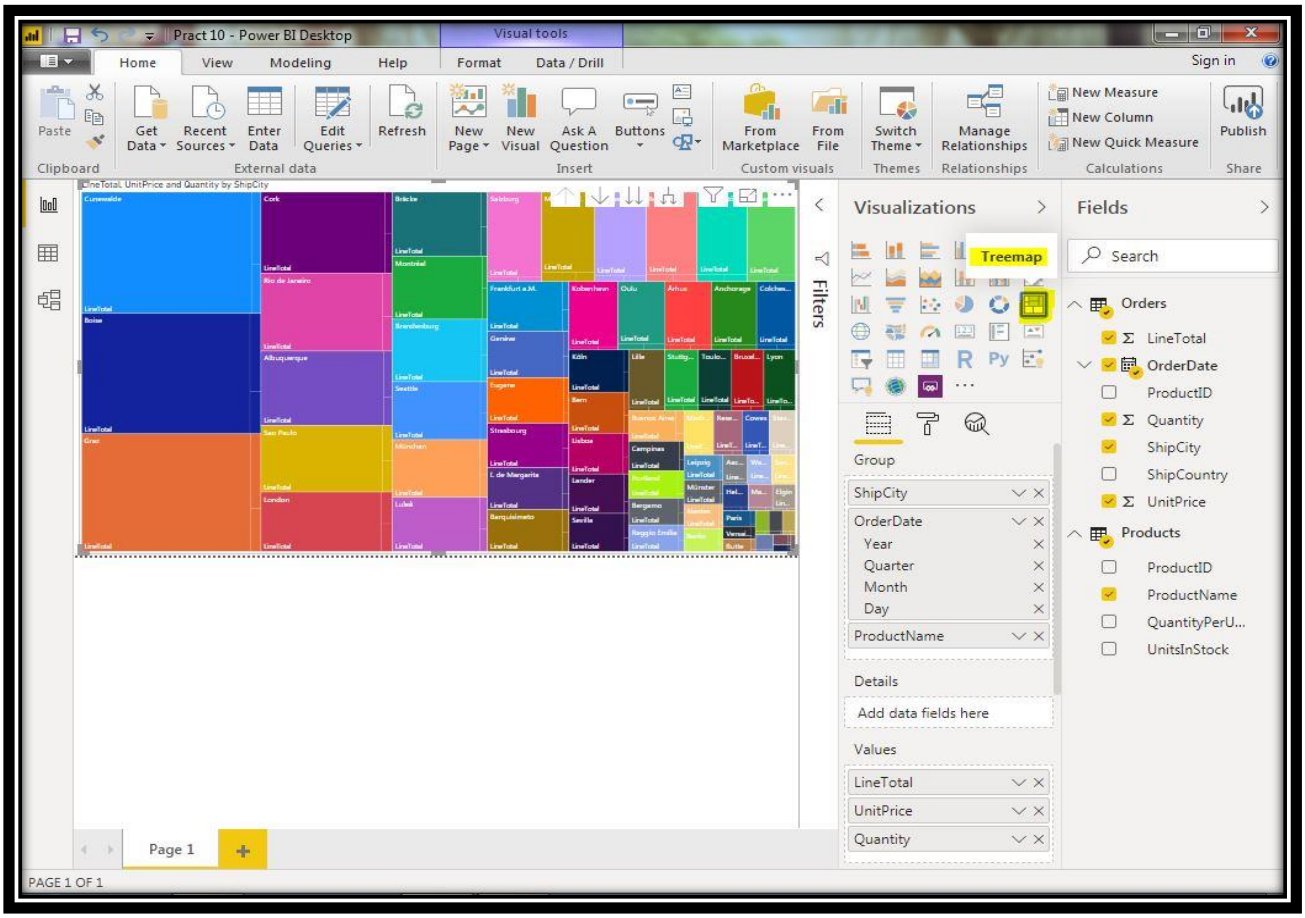


Step 2: Interact with your report visuals to analyze further



Step 3. Next, drag ShipCountry to a space on the canvas in the top right. Because you selected a geographic field, a map was created automatically. Now drag LineTotal to the Values field; the circles on the map for each country are now relative in size to the LineTotal for orders shipped to that country.


Pract 10 - Power BI Desktop
Visual tools
Home
View
Modeling
Help
Format
Data / Drill
Sign in
Paste
Get Data
Recent Sources
Enter Data
Edit Queries
Refresh
New Page
New Visual
Ask A Question
Buttons
From Marketplace
From File
Switch Theme
Manage Relationships
New Measure
New Column
New Quick Measure
Publish
Clipboard
External data
Insert
Custom visuals
Themes
Relationships
Calculations
Share
Visualizations
Map
Fields
Search
Orders
LineTotal
OrderDate
ProductID
Quantity
ShipCity
ShipCountry
UnitPrice
Products
ProductID
ProductName
QuantityPerU...
UnitsInStock
Location
ShipCity
OrderDate
Year
Quarter
Month
Day
Legend
ProductName
Latitude
Add data fields here
Longitude
Add data fields here
Size
Filters
Page 1
PAGE 1 OF 1


Pract 10 - Power BI Desktop
Visual tools
Home
View
Modeling
Help
Format
Data / Drill
Sign in
Paste
Get Data
Recent Sources
Enter Data
Edit Queries
Refresh
New Page
New Visual
Ask A Question
Buttons
From Marketplace
From File
Switch Theme
Manage Relationships
New Measure
New Column
New Quick Measure
Publish
Clipboard
External data
Insert
Custom visuals
Themes
Relationships
Calculations
Share
Visualizations
Treemap
Fields
Search
Orders
LineTotal
OrderDate
ProductID
Quantity
ShipCity
ShipCountry
UnitPrice
Products
ProductID
ProductName
QuantityPerU...
UnitsInStock
Group
ShipCity
OrderDate
Year
Quarter
Month
Day
ProductName
Details
Add data fields here
Values
LineTotal
UnitPrice
Quantity
Filters
Page 1
PAGE 1 OF 1