Keraleeya Samajam(Regd.) Dombivli's

MODEL COLLEGE

Re-Accredited Grade "A" by NAAC

Kanchan Goan Village, Khambalpada, Thakurli East – 421201
Contact No – 7045682157, 7045682158. www.model-college.edu.in

DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr. /Miss _____

*Studying in Class*_____ *Seat No.* _____

*Has completed the prescribed practicals in the subject*_____

*During the academic year*_____

Date : _____

External Examiner

Internal Examiner
M.Sc. Information Technology

INDEX

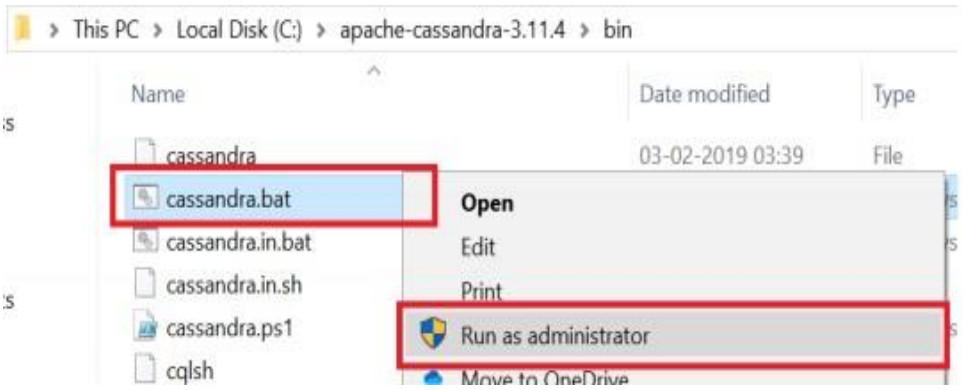
Sr. No.	Practical	Date	Signature
1	Creating Data Model using Cassandra.		
2	Conversion from different formats to HOURS format. a. XML b. JSON c. Picture (JPEG) d. Audio		
3	Utilities and Auditing		
4	Retrieving Data		
5	Assessing Data		
6	Processing Data		
7	Transforming Data		
8	Organizing Data		
9	Generating Reports		
10	Data Visualization with Power BI		

Practical 1:

Creating Data Model using Cassandra.

Go to Cassandra directory

C:\apache-cassandra-3.11.4\bin



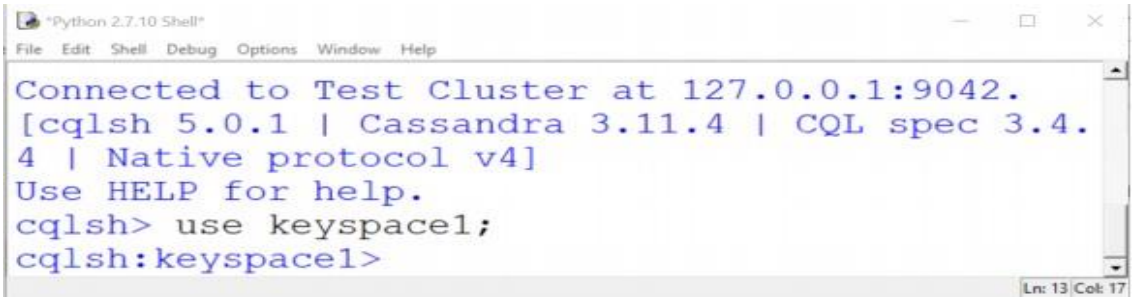
Run Cassandra.bat file

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

Creating a Keyspace using Cqlsh

Create keyspace keyspace1 with replication = {„class“:“SimpleStratergy“, „replication_factor“: 3};

Use keyspace1;



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

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

Insert into dept (dept_id, dept_name, dept_loc) values (1001, 'Accounts', 'Mumbai');

Insert into dept (dept_id, dept_name, dept_loc) values (1002, 'Marketing', 'Delhi');

Insert into dept (dept_id, dept_name, dept_loc) values (1003, 'HR', 'Chennai');

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

Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1002, 'DEFG', 1001, 'defg@company.com', '2233445566');

Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1003, 'GHIJ', 1002,

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

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

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

(6 rows)
```

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

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

(3 rows)
```

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

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

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

(3 rows)
```

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

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

(5 rows)
```

Practical 2:

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

A. XML to HORUS Format

Code :-

```
# Utility Start XML to HORUS ================================
# Standard Tools
import pandas as pd
import xml.etree.ElementTree as ET
def df2xml(data):
 header = data.columns
 root = ET.Element('root')
 for row in range(data.shape[0]):
 entry = ET.SubElement(root,'entry')
 for index in range(data.shape[1]):
 schild=str(header[index])
 child = ET.SubElement(entry, schild)
 if str(data[schild][row]) != 'nan':
 child.text = str(data[schild][row])
 else:
 child.text = 'n/a'
 entry.append(child)
 result = ET.tostring(root)
 return result
def xml2df(xml_data):
 root = ET.XML(xml_data)
 all_records = []
 for i, child in enumerate(root):
 record = {}
 for subchild in child:
 record[subchild.tag] = subchild.text
 all_records.append(record)
 return pd.DataFrame(all_records)
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()
print('================================================')
print('Input Data Values ===============================')
print('================================================')
print(InputData)
```

```
print('====================================================')
#============================================================
=
# Processing Rules ==========================================
#============================================================
=
ProcessDataXML=InputData
# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('====================================================')
print('Process Data Values ================================')
print('====================================================')
print(ProcessData)
print('====================================================')
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('====================================================')
print('XML to HORUS - Done')
print('====================================================')
# Utility done ==============================================
```

Output:

```
========== RESTART: C:\VKHCG\05-DS\9999-Data\XML2HORUS.py ==========
================================================
Input Data Values ==============================
================================================
Squeezed text (385 lines).
================================================
================================================
Process Data Values ============================
================================================
                          CountryName
CountryNumber
716                         Zimbabwe
894                           Zambia
887                            Yemen
732                   Western Sahara
876           Wallis and Futuna Islands
...                                ...
16                      American Samoa
12                             Algeria
8                              Albania
248                     Aland Islands
4                         Afghanistan

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

B. 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:

```
>>>
========== RESTART: C:\VKHCG\05-DS\9999-Data\JSON2HORUS.py ==========
Input Data Values ===================================
                          Country ISO-2-CODE ISO-3-Code  ISO-M49
0                     Afghanistan         AF         AFG        4
1                   Aland Islands         AX         ALA      248
10                      Argentina         AR         ARG       32
100                       Hungary         HU         HUN      348
101                       Iceland         IS         ISL      352
..                            ...        ...         ...      ...
95                         Guyana         GY         GUY      328
96                          Haiti         HT         HTI      332
97      Heard and Mcdonald Islands         HM         HMD      334
98   Holy See(Vatican City State)         VA         VAT      336
99                       Honduras         HN         HND      340

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

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

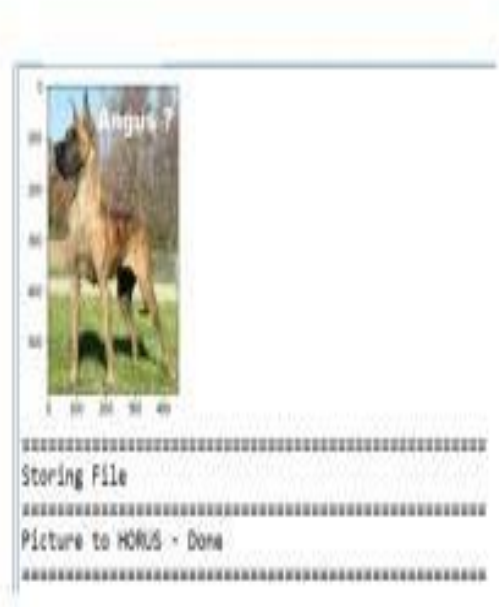
C. Picture (JPEG) to HORUS Format

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('=====================================================')
```

Output:



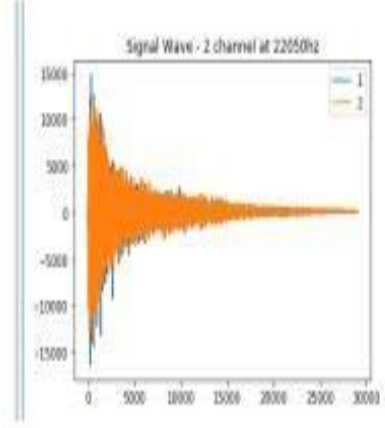
D. Audio to HORUS Format

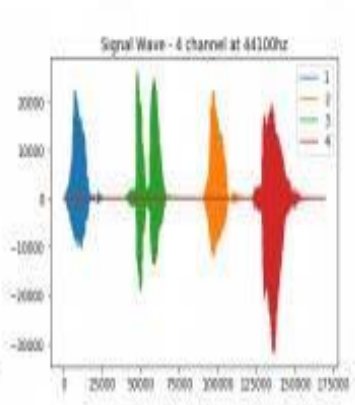
Code:

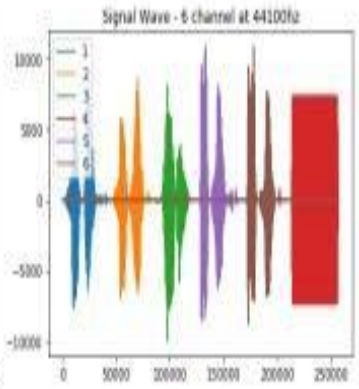
```
# Utility Start Audio to HORUS ==============================
# Standard Tools
#============================================================
from scipy.io import wavfileimport pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#============================================================
=
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'
```

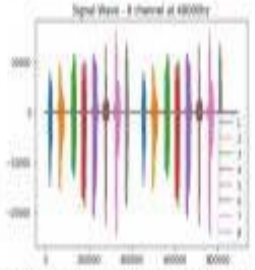
```
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('====================================================')
print('Processing : ', sInputFileName)
print('====================================================')
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')
```

Output:









```
##################################################
Audio to HDRUS - Done
##################################################
```

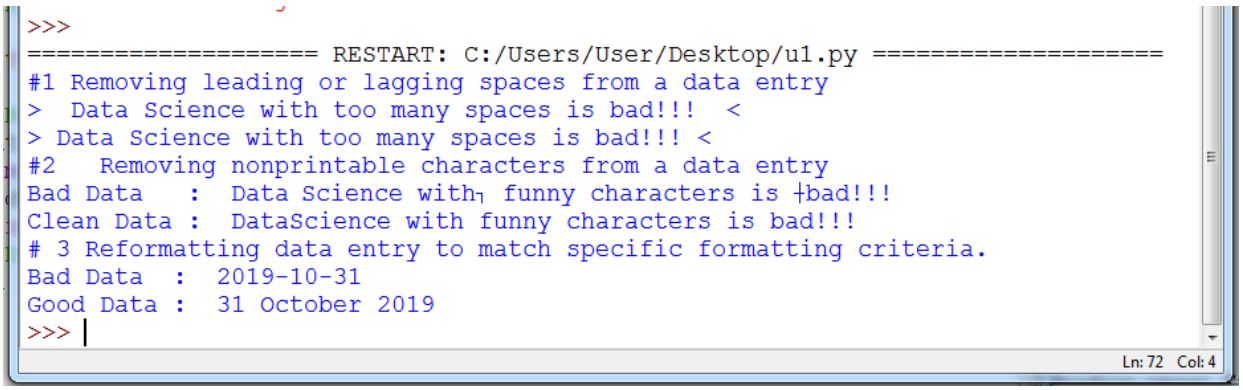
Practical 3
Utilities and Auditing

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. Averaging of Data

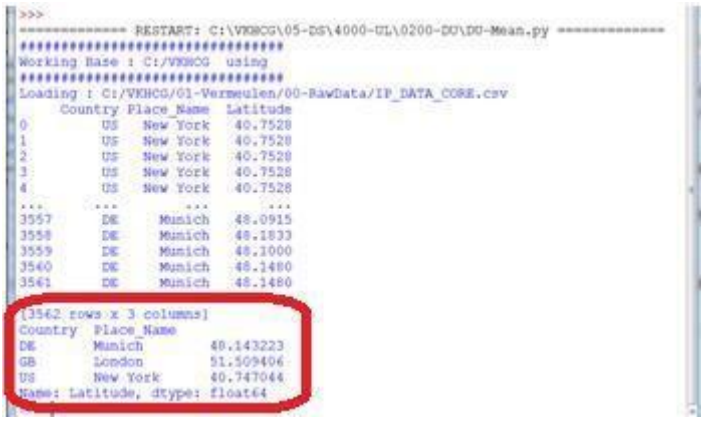
The use of averaging of features value enables the reduction of data volumes in a control fashion to improve effective data processing.

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

Code:

```
import pandas as pd
##################################################################
InputFileName='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:



Outlier Detection

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

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

Code:

```
##################################################################
# -*- coding: utf-8 -*-
##################################################################
import pandas as pd
##################################################################
InputFileName='IP_DATA_CORE.csv'
print('#################################')
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
```

```
print('################################')
print('Working Base :',Base)
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=All
Data.groupby(['
Country',
'Place_Name'])[
'Latitude'].mea
n()
StdData=AllDa
ta.groupby(['Co
untry',
'Place_Name'])[
'Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) &
(AllData.Latitude<=UpperBound)]
print(OutliersNot)
#################################################################
```

Output:

```
========== RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py
==========
################################
Working Base : C:/VKHCG
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
All Data
Country Place_Name Latitude
1910 GB London 51.5130
1911 GB London 51.5508
1912 GB London 51.5649
1913 GB London 51.5895
1914 GB London 51.5232
... ... ... ...
```

[1502 rows x 3 columns]
Outliers
Higher than 51.51263550786781
Country Place_Name Latitude
1910 GB London 51.5130

Output:

========== RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py ==========
##################################
Working Base : C:/VKHCG
##################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
All Data
Country Place_Name Latitude
1910 GB London 51.5130
1911 GB London 51.5508
1912 GB London 51.5649
1913 GB London 51.5895
1914 GB London 51.5232
[1502 rows x 3 columns]
Outliers
Higher than 51.51263550786781
Country Place_Name Latitude
1910 GB London 51.5130

C. Logging

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

C:\VKHCG\77-Yoke\Yoke_Logging.py

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 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:

```
>>>
================ RESTART: C:\VKHCG\77-Yoke\Yoke_Logging.py ================
Set up: C:/VKHCG/01-Vermeulen/01-Retrieve/Logging/Logging_61705603-bb6e-47f0-b5a
9-23d42e267311.log
root        : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-01-Retrieve-info: INFO     Practical Data Science logge
d information message.
Apllication-01-Vermeulen-01-Retrieve-warning: WARNING  Practical Data Science lo
gged a warning message.
Apllication-01-Vermeulen-01-Retrieve-error: ERROR    Practical Data Science logg
ed an error message.
Set up: C:/VKHCG/01-Vermeulen/02-Assess/Logging/Logging_a7fecb9b-4d40-474e-bc2d-
994958d85194.log
```

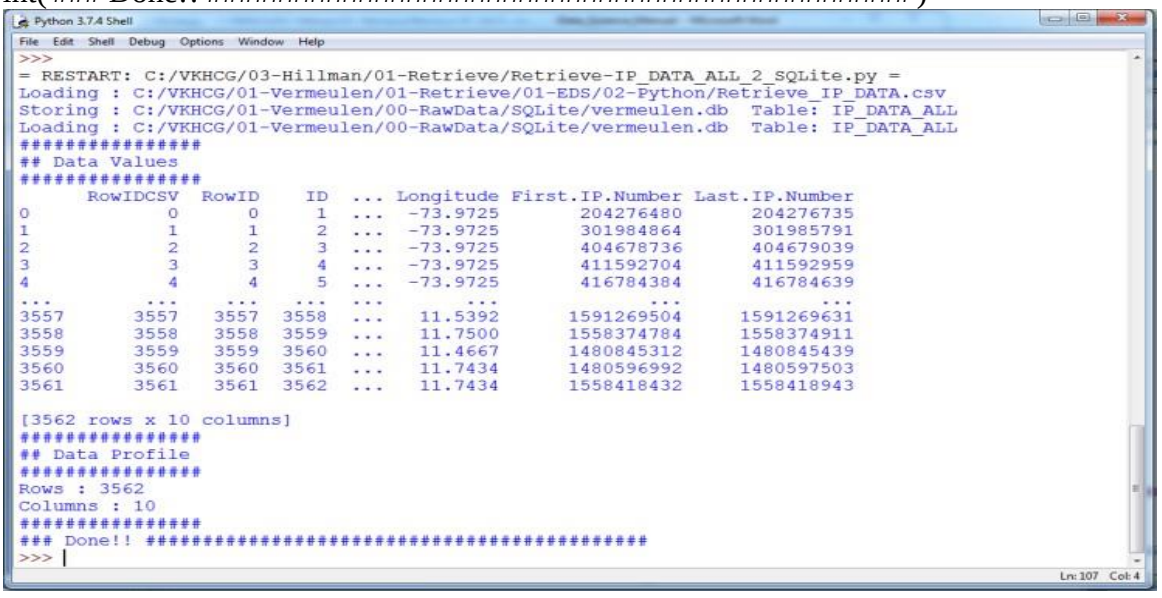
Practical 4

Retrive superstep

Connecting to other Data Sources
A. Program to connect to different data sources.

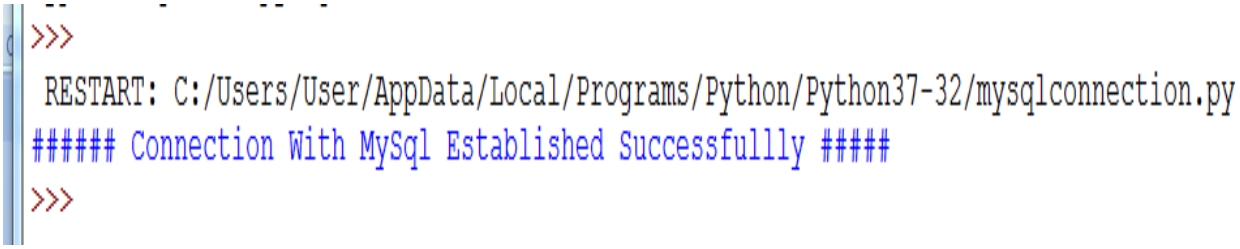
SQLite:

```
# -*- coding: utf-8 -*-
import sqlite3 as sq
import pandas as pd
Base='C:/VKHCG'
sDatabaseName=Base + '/01-Vermeulen/00-RawData/SQLite/vermeulen.db'
conn = sq.connect(sDatabaseName)
sFileName='C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-
Python/Retrieve_IP_DATA.csv'
print('Loading :',sFileName)
IP_DATA_ALL_FIX=pd.read_csv(sFileName,header=0,low_memory=False)
IP_DATA_ALL_FIX.index.names  = ['RowIDCSV']
sTable='IP_DATA_ALL'
print('Storing :',sDatabaseName,' Table:',sTable)
IP_DATA_ALL_FIX.to_sql(sTable, conn, if_exists="replace")
print('Loading :',sDatabaseName,' Table:',sTable)
TestData=pd.read_sql_query("select * from IP_DATA_ALL;", conn)
print('#################')
print('## Data Values')
print('#################')
print(TestData)
print('#################')
print('## Data Profile')
print('#################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('#################')
print('### Done!! ############################################')
```



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

```
################ Connection With MySQL ####################
import mysql.connector
conn = mysql.connector.connect(host='localhost',
database='DataScience',
user='root',
password='root')
conn.connect
if(conn.is_connected):
print('###### Connection With MySql Established Successfullly ##### ')
else:
print('Not Connected -- Check Connection Properites')
```



Microsoft Excel

```
##################Retrieve-Country-Currency.py
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import os
import pandas as pd
#####################################################################
Base='C:/VKHCG'
#####################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
#if not os.path.exists(sFileDir):
#os.makedirs(sFileDir)
#####################################################################
CurrencyRawData = pd.read_excel('C:/VKHCG/01-Vermeulen/00-
RawData/Country_Currency.xlsx')
sColumns = ['Country or territory', 'Currency', 'ISO-4217']
CurrencyData = CurrencyRawData[sColumns]
CurrencyData.rename(columns={'Country or territory': 'Country', 'ISO-4217':
'CurrencyCode'}, inplace=True)
CurrencyData.dropna(subset=['Currency'],inplace=True)
CurrencyData['Country'] = CurrencyData['Country'].map(lambda x: x.strip())
CurrencyData['Currency'] = CurrencyData['Currency'].map(lambda x:
x.strip())
CurrencyData['CurrencyCode'] = CurrencyData['CurrencyCode'].map(lambda x:
x.strip())
print(CurrencyData)
print('~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~ ')
#####################################################################
sFileName=sFileDir + '/Retrieve-Country-Currency.csv'
```

CurrencyData.to_csv(sFileName, index = False)
##

OUTPUT:

```
                        Country             Currency CurrencyCode
1                   Afghanistan        Afghan afghani          AFN
2    Akrotiri and Dhekelia (UK)         European euro          EUR
3       Aland Islands (Finland)         European euro          EUR
4                       Albania          Albanian lek          ALL
5                       Algeria        Algerian dinar          DZD
..                          ...                  ...          ...
271           Wake Island (USA)  United States dollar          USD
272  Wallis and Futuna (France)             CFP franc          XPF
274                       Yemen           Yemeni rial          YER
276                      Zambia        Zambian kwacha          ZMW
277                    Zimbabwe  United States dollar          USD

[253 rows x 3 columns]
~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~
>>>
```

PRACTICAL 05

Assessing Data

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

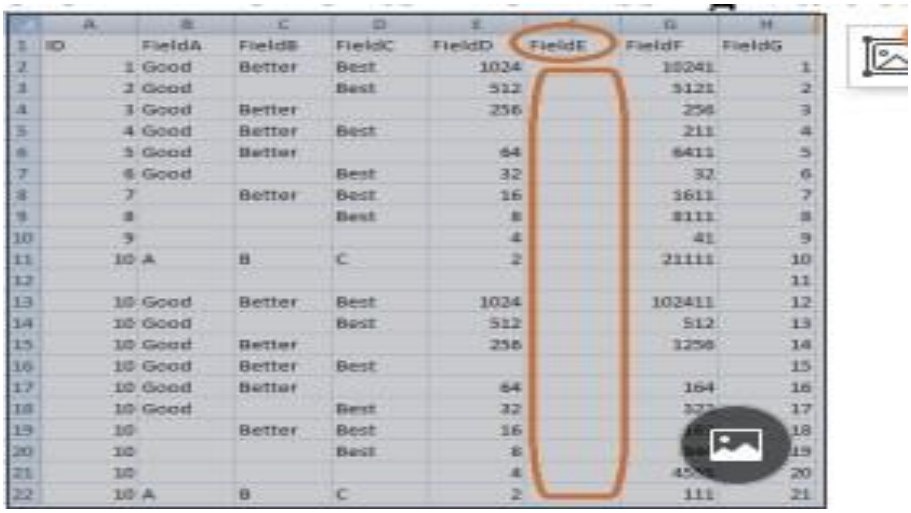
Python pandas package enables several automatic error-

management features.File Location: C:\VKHCG\01-

Vermeulen\02-Assess

Missing Values in Pandas:

i. Drop the Columns Where All Elements Are Missing Values



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

Code :

```
################### Assess-Good-Bad-01.py#########################
# -*- coding: utf-8 -*-
#############################################################
####
import
sys
import
os
import pandas as pd
#############################################################
#### Base='C:/VKHCG'
#############################################################
#### print('####################################')
print('Working Base :',Base, ' using ', sys.platform)
print('####################################')
```

```
#################################################################
####
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-01.csv'Company='01-Vermeulen'
#################################################################
#### Base='C:/VKHCG'
#################################################################
####
sFileDir=Base + '/' + Company + '/02-Assess/01-
EDS/02-Python'if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#################################################################
####
### Import Warehouse
#################################################################
####
sFileName=Base + '/' + Company + '/00-RawData/' +
sInputFileNameprint('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])
#####################################################################
sFileName=sFileDir + '/' +
sOutputFileName
TestData.to_csv(sFileName, index =
False)
#####################################################################
print('#################################')
print('### Done!! ####################')
print('#################################')
#################################################################
####
Output:
>>>
======= 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 FieldG0 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
All of column E has been deleted, owing to the fact that all values in that column were missingvalues/errors.

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

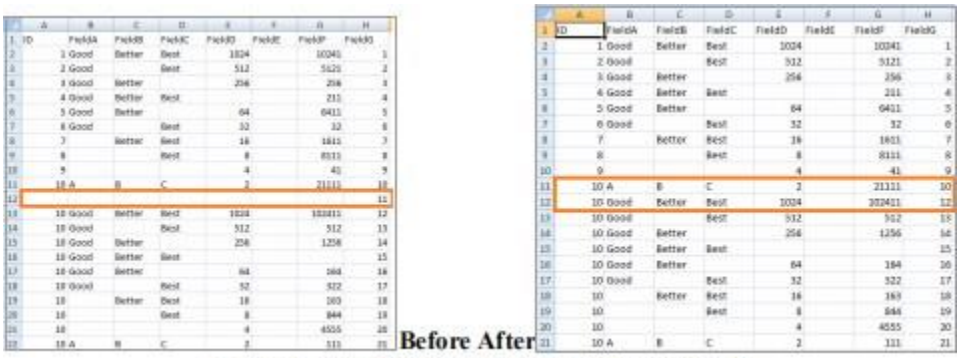
```
################## Assess-Good-Bad-02.py############################ import sys
import os
import pandas
as pd
Base='C:/VKHC
G'
sInputFileName='Good-or-
Bad.csv'
sOutputFileName='Good-or-Bad-
02.csv'Company='01-Vermeulen'
#################################################################
#### Base='C:/VKHCG'
#################################################################
####
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/' +
sInputFileNameprint('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('################################')
#############################################################
####
>>>
======= RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-Good-Bad-02.py
=======
############################
#####
Working Base : C:/VKHCG using
win32
############################
#####
Loading : C:/VKHCG/01-Vermeulen/00-
```

```
RawData/Good-or-Bad.csv
##############################
## Raw Data Values
##########################
ID FieldA FieldB FieldC
FieldD FieldE FieldF FieldG0
1.0 Good Better Best 1024.0
NaN 10241.0 1
1 2.0 Good NaN Best 512.0 NaN
5121.0 2
############################
####
## Data Profile
##########################
#####
Rows : 21
Columns : 8
##########################
#####
##########################
#####
## Test Data Values
##########################
#####
Field
G0 1
1 2
##########################
#####
## Data Profile
##########################
#####
Rows : 21
Columns : 1
############################
####
##########################
######### Done!!
####################
############################
####
>>>
iii.Keep Only the Rows That Contain a Maximum of Two
Missing Values#################### Assess-Good-Bad-
03.py ################
# -*- coding: utf-8 -*-
#############################################################
####
```

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

```
TestData=RawData.dropna(thresh=2)
print('############################
######')
print('## Test Data Values')
print('################################')
print(TestData)
print('############################
#####')
print('## Data Profile')
print('################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('############################
#####')
sFileName=sFileDir + '/' +
sOutputFileName
TestData.to_csv(sFileName, index =
False)
#####################################################################
print('################################')
print('### Done!! ####################')
print('################################')
```



Before After

Row with more than two missing values got deleted.

The next step along the route is to generate a full network routing solution for the company, toresolve the data issues in the retrieve data.

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

```
########## Assess-Network-Routing-Company.py
######################import sys
import os
import pandas as pd
```

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

pd.options.mode.chained_assignment = None

################################################################
 Base='C:/VKHCG' ################################################################
print('###################################')
print('Working Base :',Base, ' using Windows')

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

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

####

sInputFileName1='01-Retrieve/01-EDS/01-

R/Retrieve_Country_Code.csv' sInputFileName2='01-Retrieve/01-

EDS/02-Python/Retrieve_Router_Location.csv' sInputFileName3='01-

Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv'

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

#########

sOutputFileName='Assess-Network-Routing-

Company.csv' Company='01-Vermeulen'

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

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

### Import Country Data

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

###########

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

sInputFileName1

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

print('Loading :',sFileName)

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

CountryData=pd.read_csv(sFileName,header=0,low_memory=False,

encoding="latin-1")print('Loaded Country:',CountryData.columns.values)
```

```
print('################################')
############################################################
## Assess Country Data
############################################################
#### print('################################')
print('Changed :',CountryData.columns.values)
CountryData.rename(columns={'Country': 'Country_Name'},
inplace=True) CountryData.rename(columns={'ISO-2-CODE':
'Country_Code'}, inplace=True)CountryData.drop('ISO-M49',
axis=1, inplace=True)
CountryData.drop('ISO-3-Code', axis=1,
inplace=True)CountryData.drop('RowID',
axis=1, inplace=True) print('To
:',CountryData.columns.values)
print('################################
')
##########################################################
###########
##########################################################
########
### Import Company Data
############################################################
####
sFileName=Base + '/' + Company + '/' +
sInputFileName2
print('################################')
```

```
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('################################')
############################################################
####
## Assess Company Data
############################################################
#### print('################################')
print('Changed :',CompanyData.columns.values)
CompanyData.rename(columns={'Country': 'Country_Code'},
inplace=True)print('To :',CompanyData.columns.values)
print('################################')
### Import Customer Data
############################################################
####
sFileName=Base + '/' + Company + '/' +
sInputFileName3
print('################################')
print('Loading :',sFileName)
print('################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_me
mory=False, encoding="latin-
1")print('################################')
```

```
print('Loaded Customer :',CustomerRawData.columns.values)
print('################################')
############################################################
####
CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('################################')
print('Remove Blank Country Code')
print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])
print('################################')
#####################################################################
print('################################')
print('Changed :',CustomerData.columns.values)
CustomerData.rename(columns={'Country': 'Country_Code'},
inplace=True) print('To :',CustomerData.columns.values)
print('################################')
############################################################
###### print('################################')
print('Merge Company and Country Data')
print('################################')
CompanyNetworkData=pd.mer
ge(CompanyData,
CountryData,
how='inner',
on='Country_C
ode'
)
##############################################
```

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

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

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

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

####
```

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

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

Next, Access the the customers location using network router location.

```
####################Assess-Network-Routing-Customer.py
###################### import sys
import os
import pandas as pd
############################################################
####
pd.options.mode.chained_assignment = None
############################################################
#### Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
############################################################
#######
sInputFileName=Base+'/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network_Routing-
Customer.csv' ######################################################################
sOutputFileName='Assess-Network-Routing-
Customer.gml' Company='01-Vermeulen'
######################################################################
# Import Country Data
sFileName=sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
CustomerData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")print('Loaded Country:',CustomerData.columns.values)
print('#################################')
print(CustomerData.head())
print('#################################')
```

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

Output

Assess-Network-Routing-Customer.csv

	A	B	C	D	E
1	ue_Countr	tomer_Place_Na	Customer_Latitude	Customer_Longitude	Customer_Country_Name
2	BW	Gaborone	-24.6464	25.9119	Botswana
3	BW	Francistown	-21.1667	27.5167	Botswana
4	BW	Maun	-19.9833	23.4167	Botswana
5	BW	Molepolole	-24.4167	25.5333	Botswana
6	NE	Niamey	13.5167	2.1167	Niger
7	MZ	Maputo	-25.9653	32.5892	Mozambique
8	MZ	Tete	-16.1564	33.5867	Mozambique
9	MZ	Quelimane	-17.8708	36.8883	Mozambique
10	MZ	Chimoio	-19.1164	33.4833	Mozambique
11	MZ	Matola	-25.9622	32.4589	Mozambique
12	MZ	Pemba	-12.9708	40.5078	Mozambique
13	MZ	Lichinga	-13.3128	35.2406	Mozambique
14	MZ	Maxixe	-23.8597	35.3472	Mozambique
15	MZ	Chibuto	-24.6867	33.5308	Mozambique
16	MZ	Pessane Garcia	-25.4428	33.9853	Mozambique
17	GH	Tema	5.6167	-0.0167	Ghana
18	GH	Kumasi	6.6833	-1.6167	Ghana
19	GH	Takoradi	4.8833	-1.75	Ghana
20	GH	Accra	5.55	-0.2167	Ghana

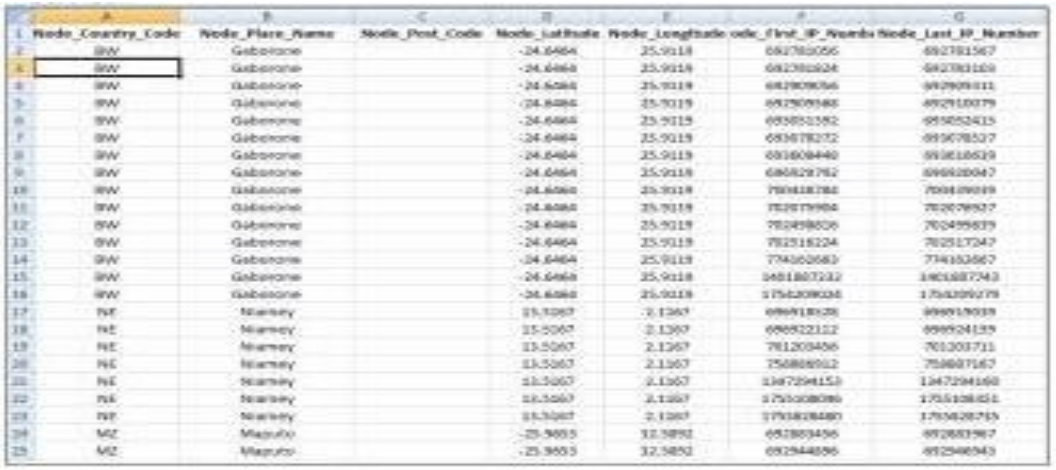
Assess-Network-Routing-Node.py

```
##################################################################
####
import
sys
import
os
import pandas as pd
##################################################################
####
pd.options.mode.chained_assignment = None
##################################################################
#### Base='C:/VKHCG'
##################################################################
#### print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
##################################################################
####
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
##################################################################
sOutputFileName='Assess-Network-Routing-
Node.csv' Company='01-Vermeulen'
##################################################################
### Import IP Data
##################################################################
####
sFileName=Base + '/' + Company + '/' +
sInputFileName
```

```
print('################################')
print('Loading :',sFileName)
print('##########################
######')
IPData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")print('Loaded IP :', IPData.columns.values)
print('################################')
#########################################################
######## print('################################')
print('Changed
:',IPData.columns.values)
IPData.drop('RowID', axis=1,
inplace=True)IPData.drop('ID',
axis=1, inplace=True)
IPData.rename(columns={'Country': 'Country_Code'},
inplace=True) IPData.rename(columns={'Place.Name':
'Place_Name'}, inplace=True)
IPData.rename(columns={'Post.Code': 'Post_Code'},
inplace=True) IPData.rename(columns={'First.IP.Number':
'First_IP_Number'}, inplace=True)
IPData.rename(columns={'Last.IP.Number': 'Last_IP_Number'},
inplace=True)print('To :',IPData.columns.values)
print('################################')
#########################################################
###########print('################################')
print('Change
',IPData.columns.values)for i in
IPData.columns.values:
j='Node_'+i
IPData.rename(columns={i: j},
inplace=True) print('To ',
IPData.columns.values)
print('##############################
#')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-
EDS/02-Python'if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#############################################################
####
sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('##########################
######')
IPData.to_csv(sFileName, index = False, encoding="latin-1")
#####################################################################
print('################################')
```

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

Output:
C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing_Node.csv



	A	B	C	D	E	F	G
1	Node_Country_Code	Node_Place_Name	Node_Post_Code	Node_Latitude	Node_Longitude	ode_First_IP_Numbe	Node_Last_IP_Number
2	BW	Gaborone		-24.6464	25.9119	692760056	692781567
3	BW	Gaborone		-24.6464	25.9119	692781624	692781103
4	BW	Gaborone		-24.6464	25.9119	692809056	692809111
5	BW	Gaborone		-24.6464	25.9119	692809568	692910079
6	BW	Gaborone		-24.6464	25.9119	695852592	695852415
7	BW	Gaborone		-24.6464	25.9119	693678272	693678527
8	BW	Gaborone		-24.6464	25.9119	693808448	693816639
9	BW	Gaborone		-24.6464	25.9119	695828752	696920047
10	BW	Gaborone		-24.6464	25.9119	700418784	700419039
11	BW	Gaborone		-24.6464	25.9119	702979984	702979927
12	BW	Gaborone		-24.6464	25.9119	702498624	702499839
13	BW	Gaborone		-24.6464	25.9119	702518224	702517247
14	BW	Gaborone		-24.6464	25.9119	734162687	734163967
15	BW	Gaborone		-24.6464	25.9119	1481807232	1481887743
16	BW	Gaborone		-24.6464	25.9119	1756209024	1756209279
17	NE	Niamey		13.5067	2.1267	696918528	696919039
18	NE	Niamey		13.5067	2.1267	696922112	696924159
19	NE	Niamey		13.5067	2.1267	761203456	761203711
20	NE	Niamey		13.5067	2.1267	756869632	756887167
21	NE	Niamey		13.5067	2.1267	1347284153	1347284160
22	NE	Niamey		13.5067	2.1267	1755108096	1755108351
23	NE	Niamey		13.5067	2.1267	1793828480	1793828735
24	MZ	Maputo		-25.9655	32.5892	682869456	682883967
25	MZ	Maputo		-25.9655	32.5892	692944896	692946943

Practical 6:

Processing Data

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

Open your Python editor and create a file named Process_Time.py. Save it into directoryC:\VKHCG\01-Vermeulen\03-Process.

```
import
sys
import
os
from datetime import
datetime from datetime
import timedelta
from pytz import timezone,
all_timezonesimport pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
pd.options.mode.chained_assignment
= Noneif 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='VehicleDa
ta.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','DateTimeVa
lue']] 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+=1z+=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'],inpl
ace=False) sZone=zone.replace('/','-').replace(' ','')
sTable = 'Process-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn1,
if_exists="replace")sTable = 'Satellite-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn2,
if_exists="replace")print('#################')
print('Vacuum
Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#################')
print('### Done!! ############################################')
```

Practical 07

Transform utility

Simple Linear Regression
Linear regression is used if there is a relationship or significant association between the variables. This can be checked by scatterplots. If no linear association appears between the variables, fitting a linear regression model to the data will not provide a useful model.A linear regression line has equations in the following form:
Y = a + bX,
Where, X = explanatory variable and

Y = dependent
variableb = slope
of the line
a = intercept (the value of y when x = 0)

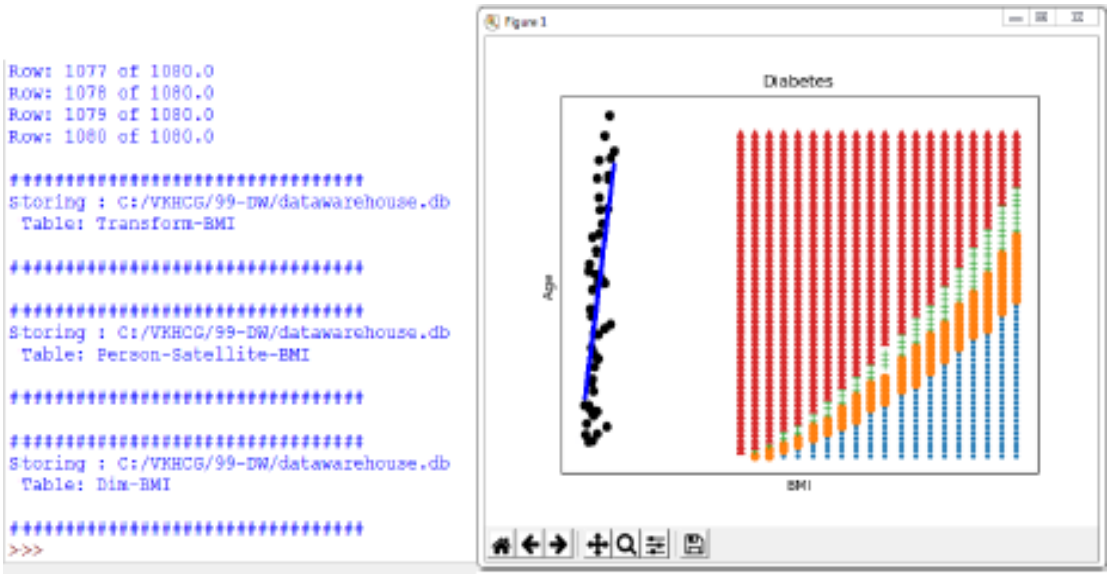
```
#############################################################
####
# -*- coding: utf-8 -*-
#############################################################
####
import
sys
import
os
import pandas as pd
import sqlite3 as sq
import matplotlib.pyplot
as pltimport numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
#############################################################
#### Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#############################################################
########
#############################################################
###########
Company='01-Vermeulen'
#############################################################
####
sDataBaseDir=Base + '/' + Company + '/04-
Transform/SQLite'if not
os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir +
'/Vermeulen.db'conn1 =
sq.connect(sDatabaseName)
#################################################################
```

```
sDataVaultDir=Base + '/88-DV'
if not
os.path.exists(sDataVaultDir):
os.makedirs(sDataVaultDir)
#####################################################################
sDatabaseName=sDataVaultDir +
'/datavault.db'conn2 =
sq.connect(sDatabaseName)
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not
os.path.exists(sDataWarehouseDir
):
os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir +
'/datawarehouse.db'conn3 =
sq.connect(sDatabaseName)
#####################################################################
t=0
tMax=((300-100)/10)*((300-30)/5)
for heightSelect in
range(100,300,10):for
weightSelect in
range(30,300,5): height =
round(heightSelect/100,3)
weight = int(weightSelect)
bmi =
weight/(height*height)if
bmi <= 18.5:
BMI_Result=1
elif bmi > 18.5 and bmi < 25:
BMI_Result=2
elif bmi > 25 and bmi < 30:
BMI_Result
=3elif bmi >
30:
BMI_Result=4
else:
BMI_Result=0
PersonLine=[('PersonID', [str(t)]),
('Height', [height]),
('Weight', [weight]),
('bmi', [bmi]),
('Indicator', [BMI_Result])]
t+=1
print('Row:',t,'of',tMax)
if t==1:

PersonFrame = pd.DataFrame.from_items(PersonLine)
else:
PersonRow = pd.DataFrame.from_items(PersonLine)
PersonFrame = PersonFrame.append(PersonRow)
#####################################################
#########
```

```
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#################################################################
####
sTable = 'Transform-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n
Table:',sTable)
print('\n############################
######')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
#################################################################
#################################################################
sTable = 'Person-Satellite-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n
Table:',sTable)
print('\n############################
######')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
#################################################################
sTable = 'Dim-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n
Table:',sTable)
print('\n############################
######')
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
#################################################################
fig = plt.figure()
PlotPerson=DimPerson[DimPerson['Indicator']==1
] x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, ".")
PlotPerson=DimPerson[DimPerson['Indicator']==2
]x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "o")
PlotPerson=DimPerson[DimPerson['Indicator']==3
] x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "+")
PlotPerson=DimPerson[DimPerson['Indicator']==4
]x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "^")
plt.axis('tight')
plt.title("BMI Curve")
plt.xlabel("Height(met
ers)")
plt.ylabel("Weight(kg)
")
```

```
plt.plot()
# Load the diabetes dataset
diabetes =
datasets.load_diabetes()#
Use only one feature
diabetes_X = diabetes.data[:,
np.newaxis, 2]diabetes_X_train =
diabetes_X[:-30] diabetes_X_test =
diabetes_X[-50:] diabetes_y_train =
diabetes.target[:-30] diabetes_y_test
= diabetes.target[-50:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train,
diabetes_y_train) diabetes_y_pred =
regr.predict(diabetes_X_test)


print('Coefficients: \n',
regr.coef_)print("Mean
squared error: %.2f"
% mean_squared_error(diabetes_y_test, diabetes_y_pred))
print('Variance score: %.2f' % r2_score(diabetes_y_test,
diabetes_y_pred))plt.scatter(diabetes_X_test, diabetes_y_test,
color='black') plt.plot(diabetes_X_test, diabetes_y_pred,
color='blue', linewidth=3) plt.xticks(())
plt.yticks(())
plt.axis('tight')
plt.title("Diabet
es")
plt.xlabel("BMI
")
plt.ylabel("Age"
) plt.show()
```

Output:



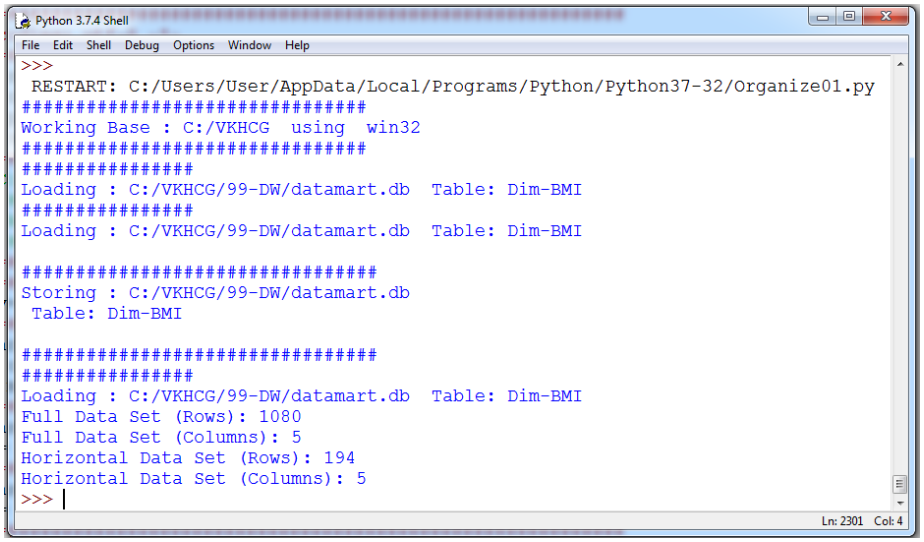
Practical 8:

Organizing Data

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

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

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



Vertical Style

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

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
#################################################################
Company='01-Vermeulen'
#################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir):
################################
################################
```

```
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#################################################################
print('##################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
        ###################################################################

print('##################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
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:

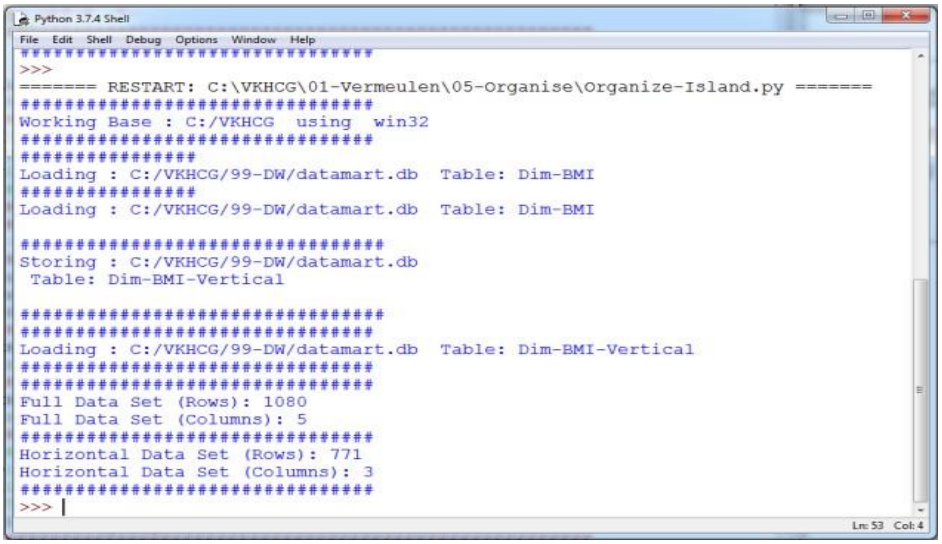
Island Style

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

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
#################################################################

Company='01-Vermeulen'
#################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
#################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
Height,\
Weight,\
```

```
Indicator\
FROM [Dim-BMI]\
WHERE Indicator > 2\
ORDER BY \
Height,\
Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####################################################################
sTable = 'Dim-BMI-Vertical'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('#################################')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#################################')
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#####################################################################
print('#################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('#################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('#################################')
#####################################################################
```

Output:



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

```
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
################################################################
Company='01-Vermeulen'
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
Height,\
Weight,\
Indicator,\
CASE Indicator\
WHEN 1 THEN 'Pip'\
WHEN 2 THEN 'Norman'\
WHEN 3 THEN 'Grant'\
ELSE 'Sam'\
END AS Name\
FROM [Dim-BMI]\
WHERE Indicator > 2\
ORDER BY \
Height,\
Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
################################################################
DimPerson=PersonFrame1
```

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

Output:

```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
>>>
==== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Secure-Vault.py ====
################################
Working Base : C:/VKHCG  using  win32
################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

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

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

Practical 9
Report Superstep

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

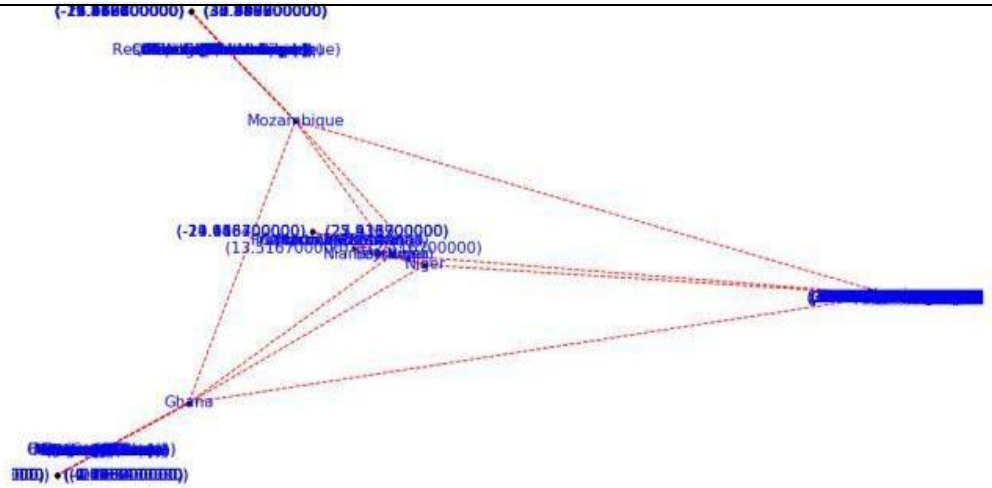
Vermeulen PLC

Vermeulen requires a map of all their customers‟ data links. Can you provide a report to deliver this? I will guide you through an example that delivers this requirement.

C:\VKHCG\01-Vermeulen\06-Report\Raport-Network-Routing- Customer.py

```
################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
################################################################
pd.options.mode.chained_assignment = None
################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + 'VKHCG'
else:
Base='C:/VKHCG'
################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv'
################################################################
sOutputFileName1='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml'
sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png'
Company='01-Vermeulen'
################################################################
################################################################
### Import Country Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
CustomerDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
CustomerData=CustomerDataRaw.head(100)
print('Loaded Country:',CustomerData.columns.values)
```

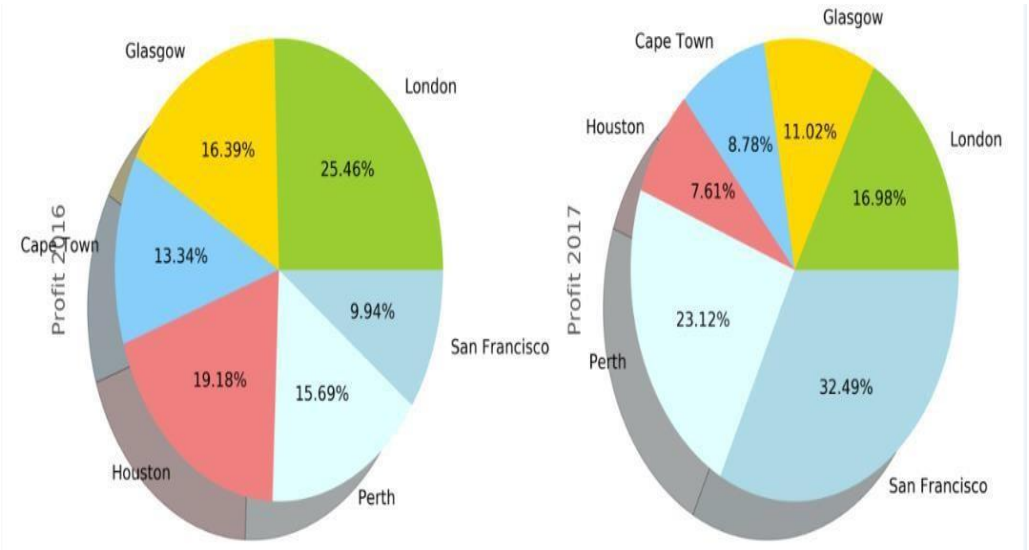
```
print('#################################')
#####################################################################
print(CustomerData.head())
print(CustomerData.shape)
#####################################################################
G=nx.Graph()
for i in range(CustomerData.shape[0]):
for j in 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('#################################')
```



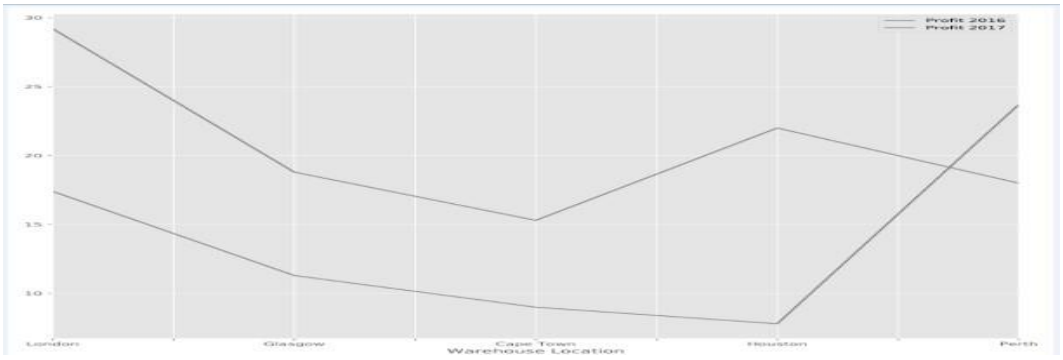
Report graph A

Graphics

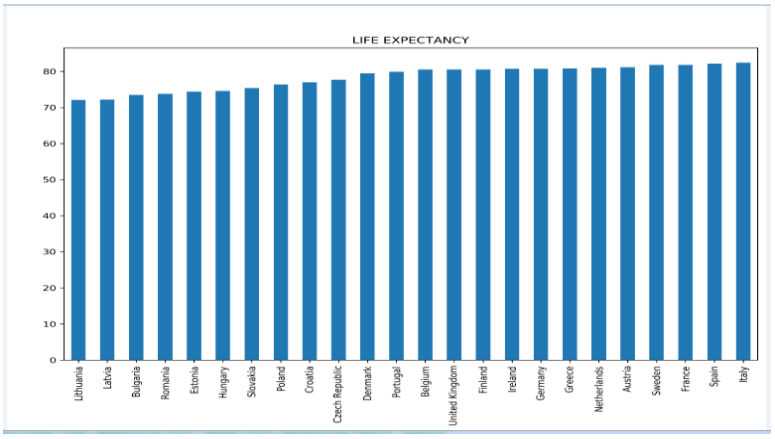
This section will now guide you through a number of visualizations that particularly useful in presenting data to my customers.
Pie Graph
Double Pie
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py



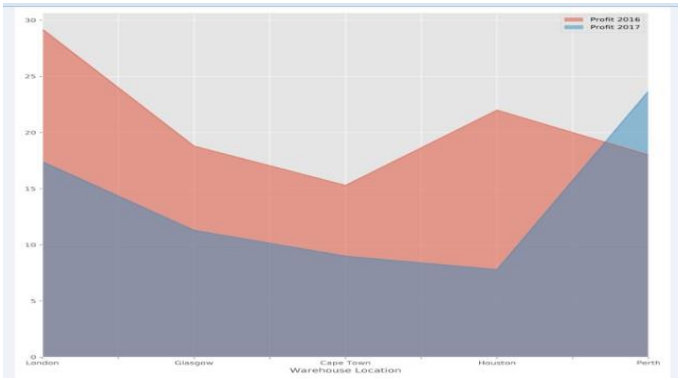
Line Graph
C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py



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

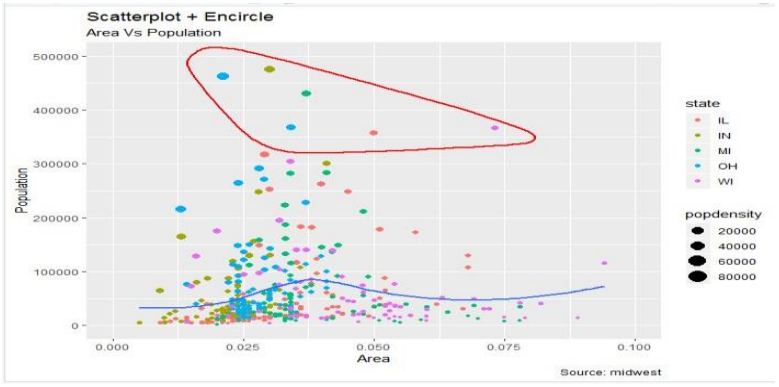


Area Graph
C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py

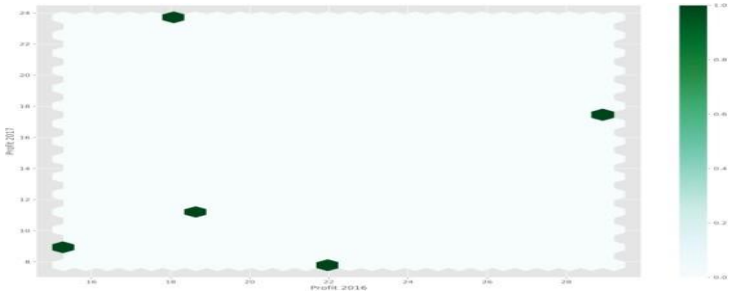


SCATTER GRAPH

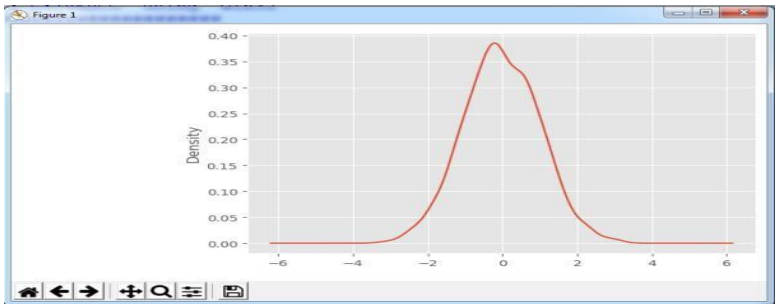
C:/ VKHCG/03-HILLMAN/06-REPORT/REPORT-SCATTERPLOT-WITH-ENCIRCLING.R



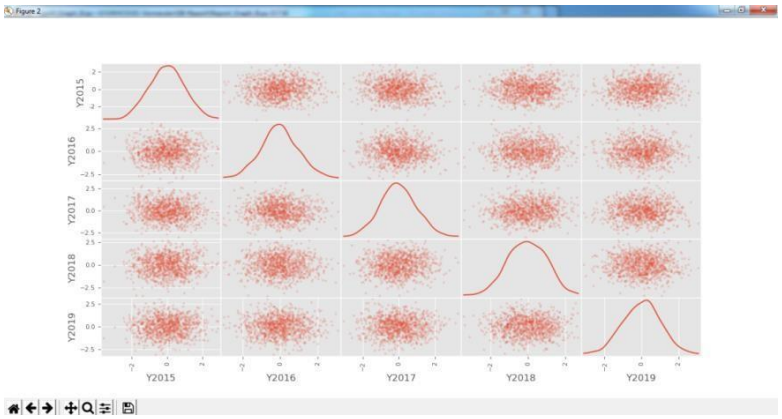
Hexbin:
Program : C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py



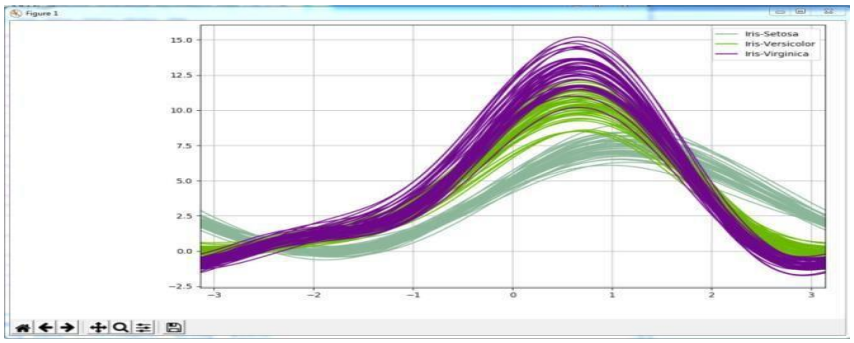
Kernel Density Estimation (KDE) Graph
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py



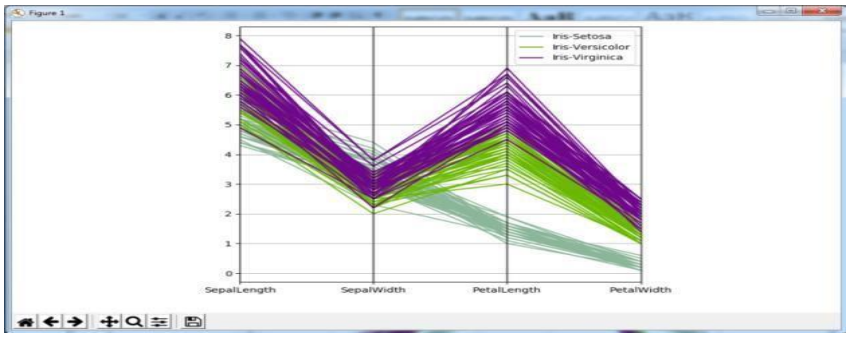
Scatter Matrix Graph
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py



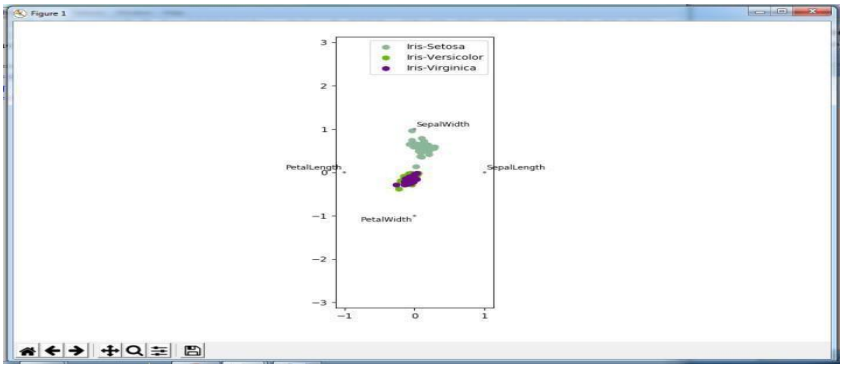
Andrews' Curves
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py



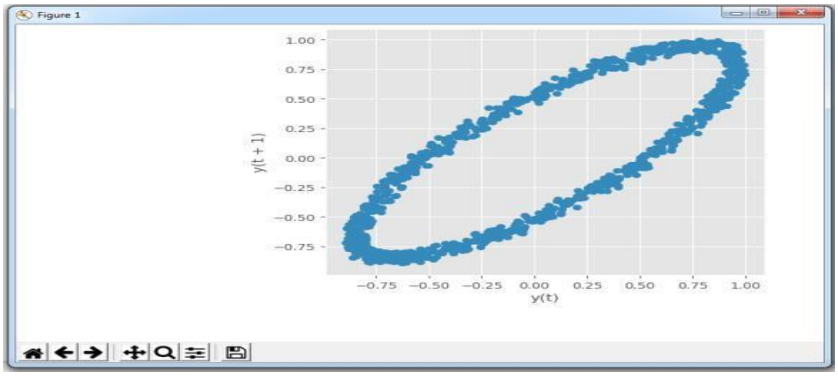
Parallel Coordinates
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py



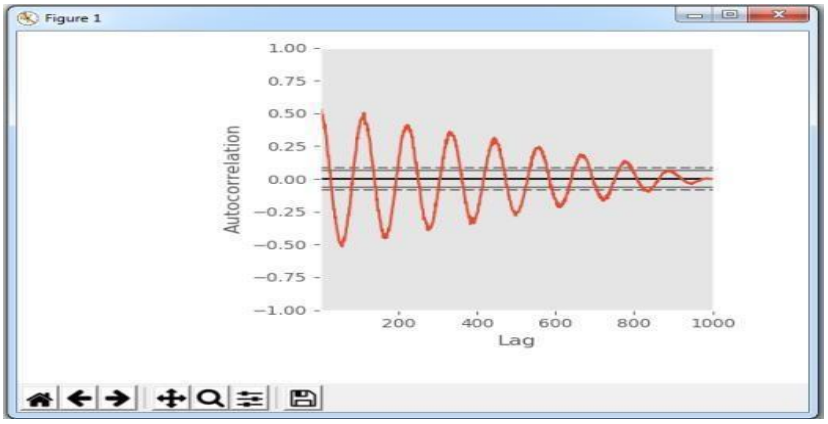
RADVIZ Method
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py



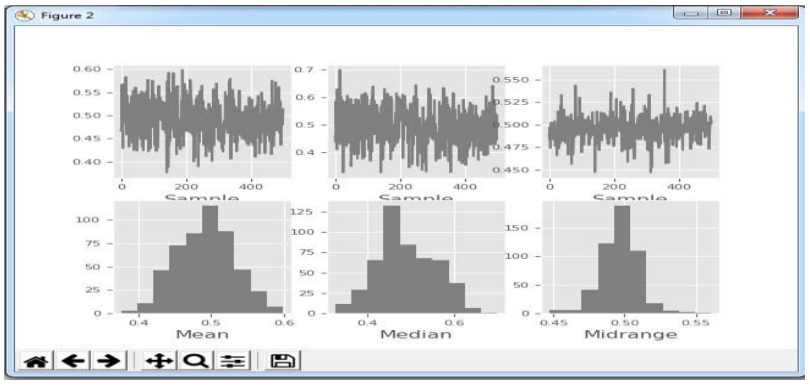
Lag Plot
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py



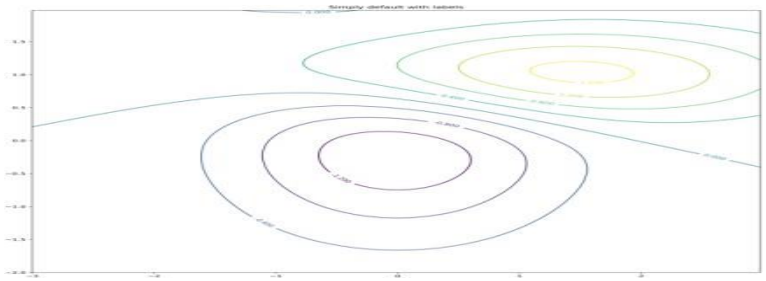
Autocorrelation Plot
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py



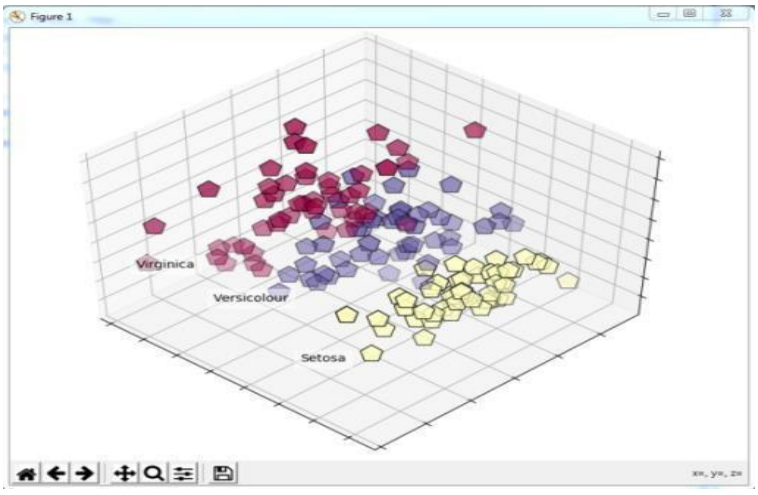
Bootstrap Plot
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py



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



3D Graphs
C:\VKHCG\01-Vermeulen\06-Report\Report_PCA_IRIS.py



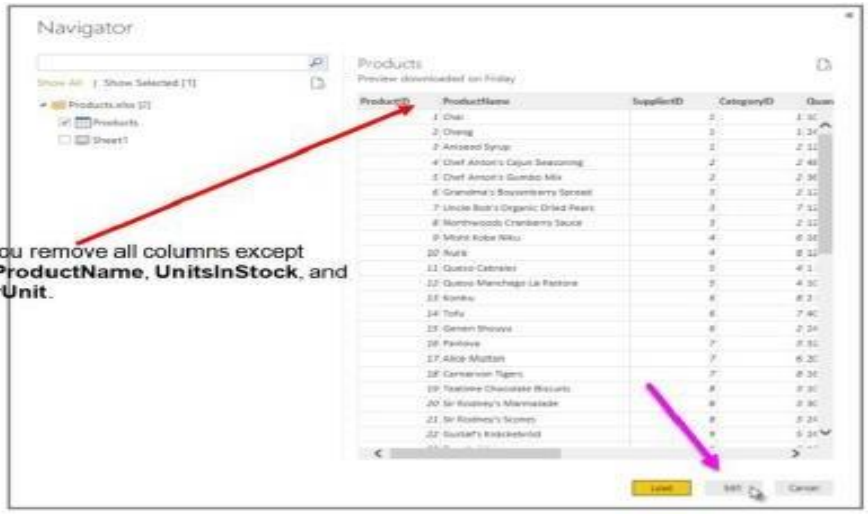
Practical 10
Data Visualization with Power BI

Case Study : Sales Data

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 it 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 **Products.xlsx** file.



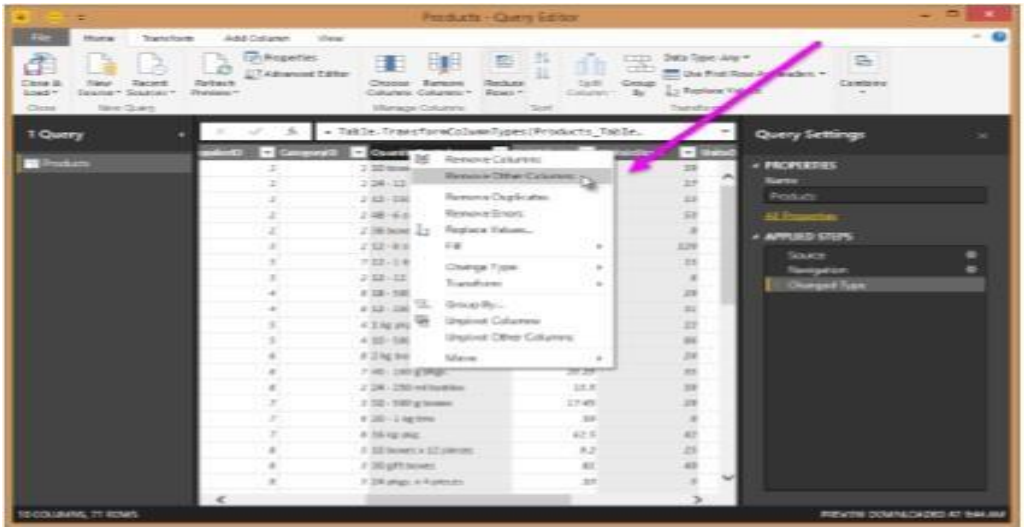
In this step you remove all columns except **ProductID**, **ProductName**, **UnitsInStock**, and **QuantityPerUnit**.

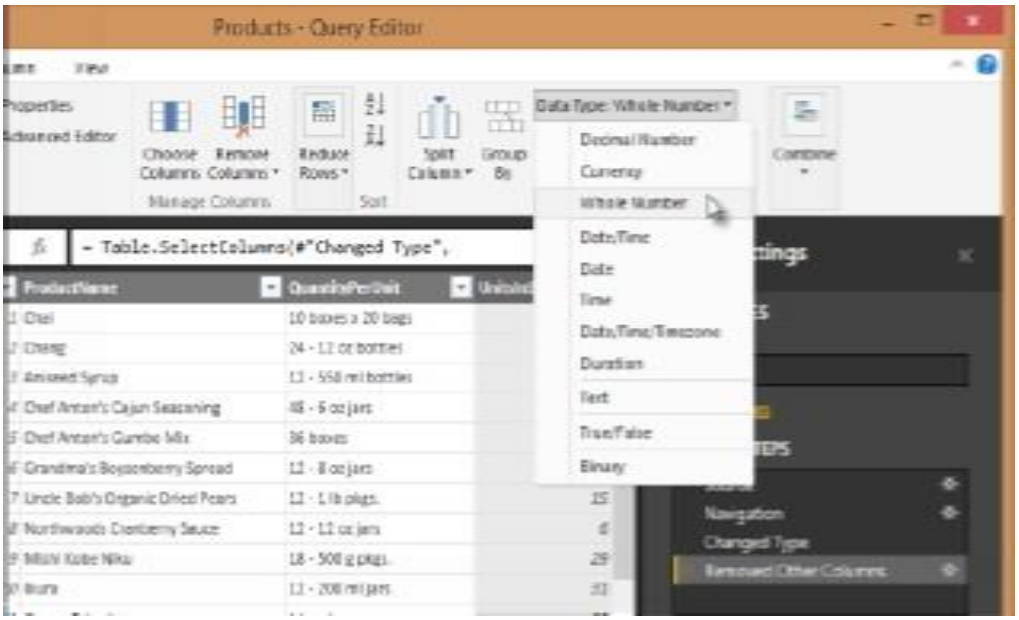
You can also open the Query Editor by selecting Edit Queries from the Home ribbon in Power BI Desktop. The following steps are performed in Query Editor.

1. In Query Editor, select the ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns

(use Ctrl+Click to select more than one column, or Shift+Click to select columns that are beside each other)

2. Select Remove Columns Remove Other Columns from the ribbon, or right-click on a column header and click Remove Other Columns.





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

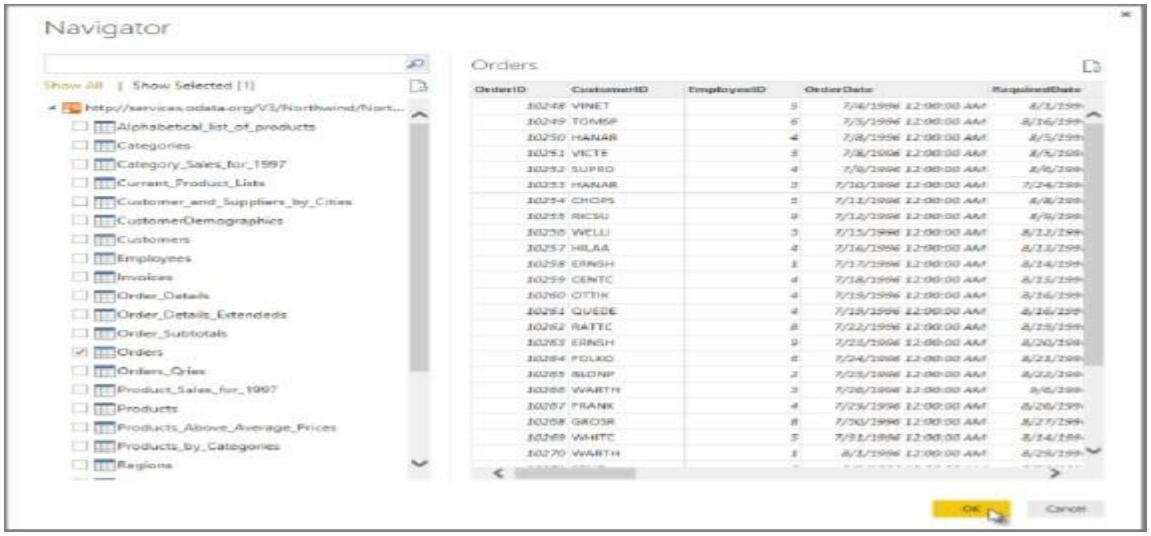
(the Data Type: button also displays the data type for the current selection).

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

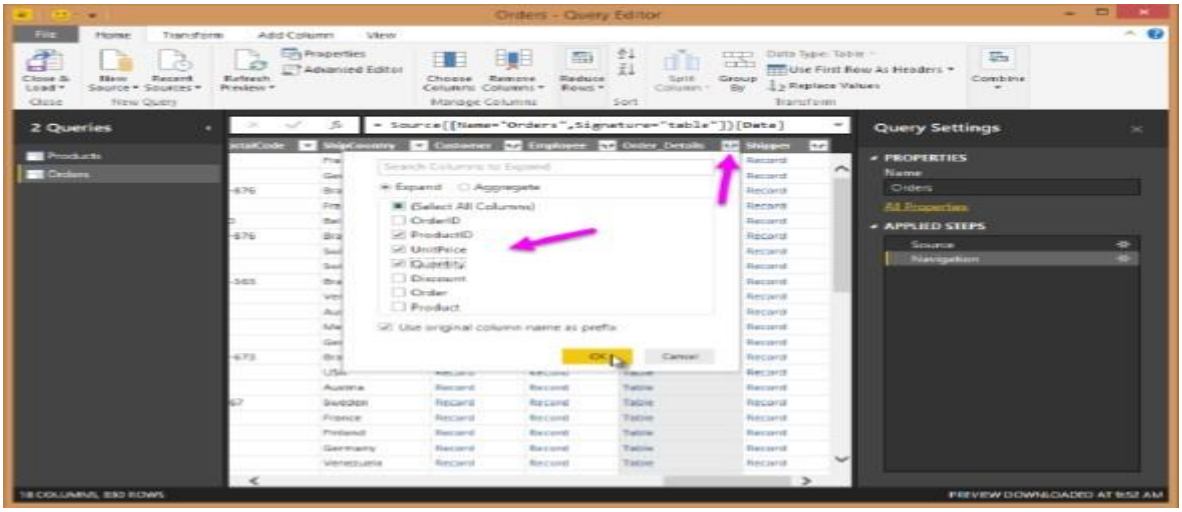
1. From the **Home** ribbon tab in Query Editor, select **Get Data.**
2. Browse to the **OData Feed** data source.
3. In the **OData Feed** dialog box, paste the **URL** for the Northwind OData feed.
4. Select OK.



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

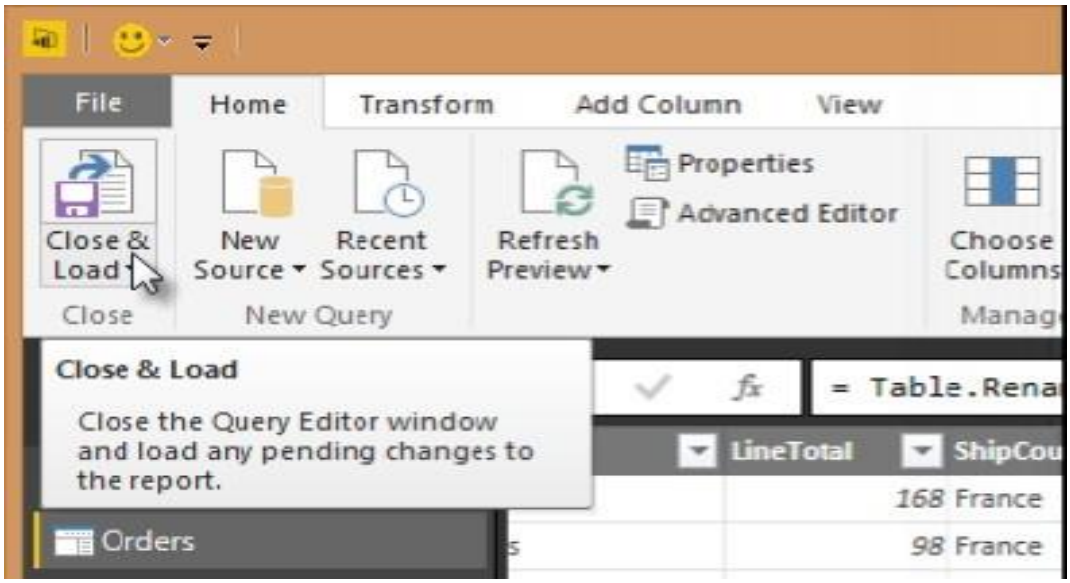
Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a **Custom Column** to calculate the line total for each **Order_Details** row.

Calculate the line total for each **Order_Details** row:

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

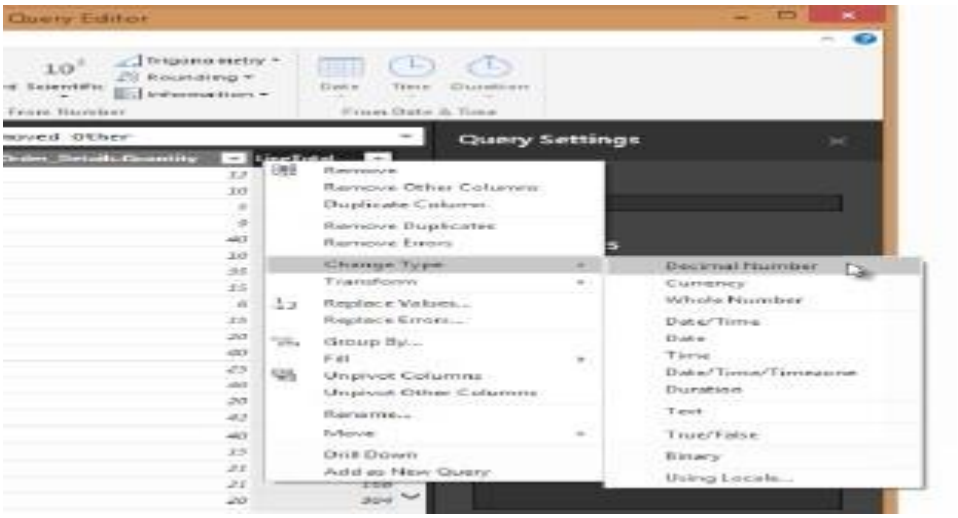
2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter **[Order_Details.UnitPrice] * [Order_Details.Quantity].**

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



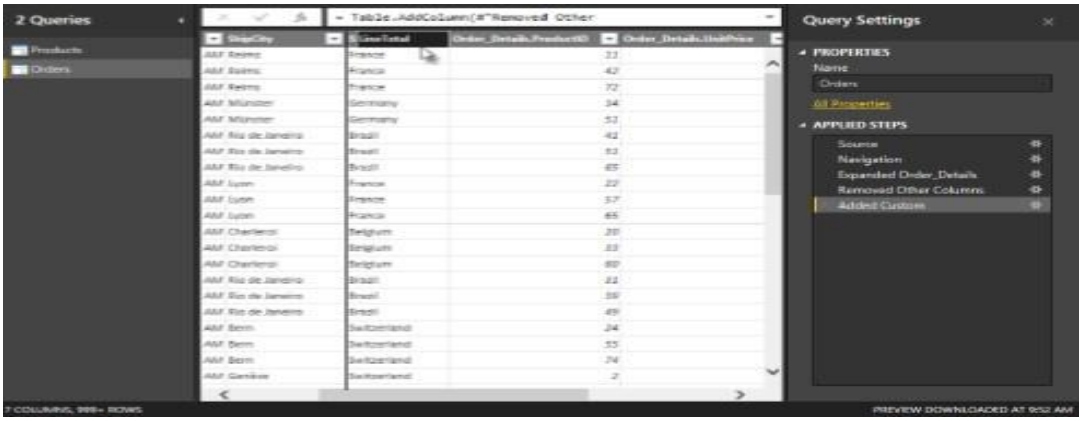
Step 5: Set the datatype of the LineTotal field

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

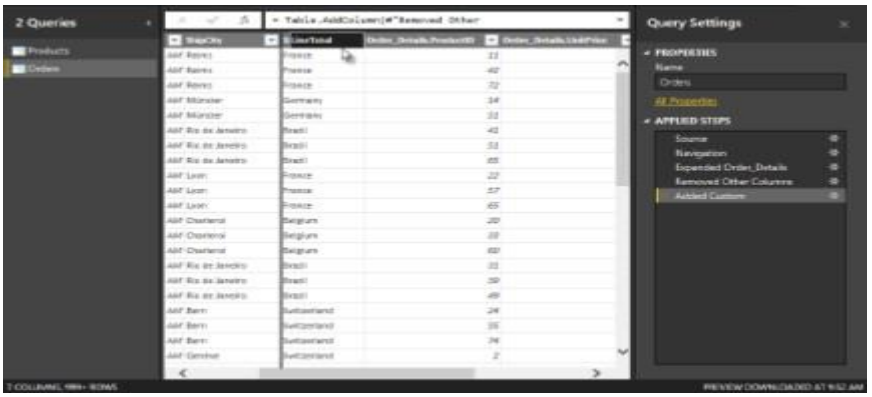


Step 6: Rename and reorder columns in the query

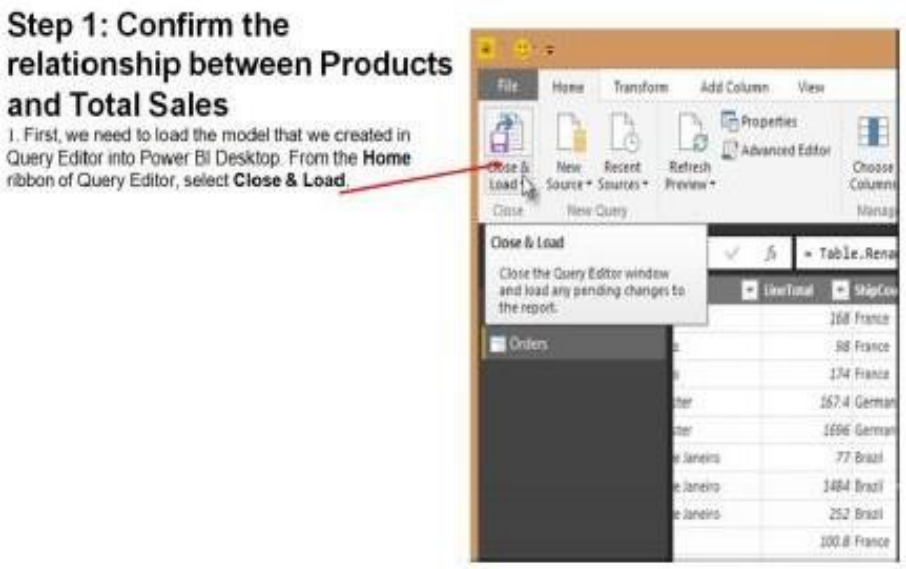
1. In **Query Editor**, drag the **LineTotal** column to the left, after **ShipCountry**.
2. Remove



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.



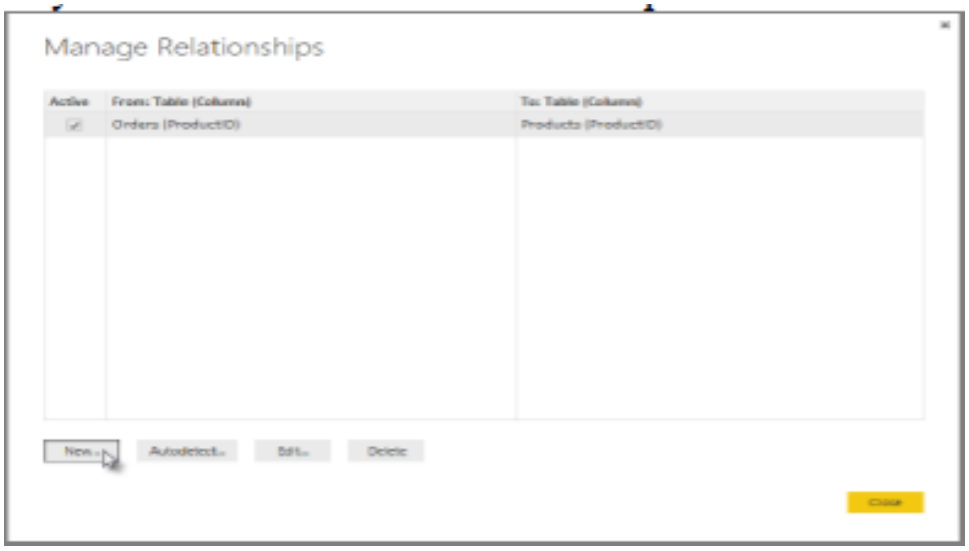
Task 3: Combine the Products and Total Sales queries

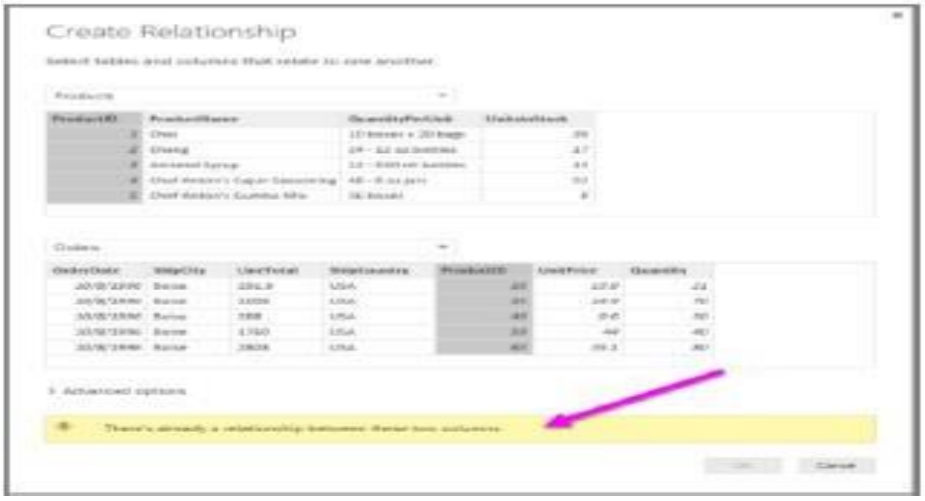


2. Power BI Desktop loads the data from the two queries
3. Once the data is loaded, select the Manage Relationships button Home ribbon

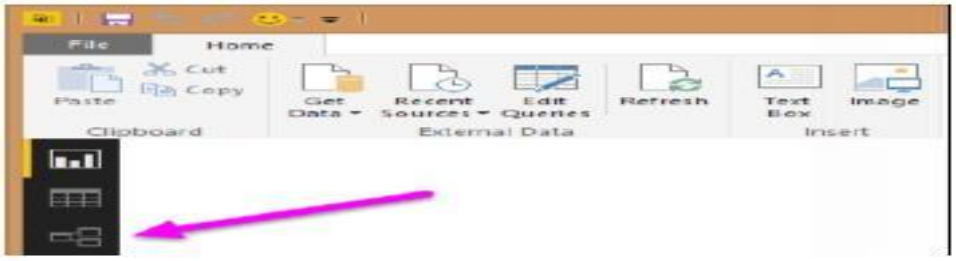
4. Select the New… button

5. 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.



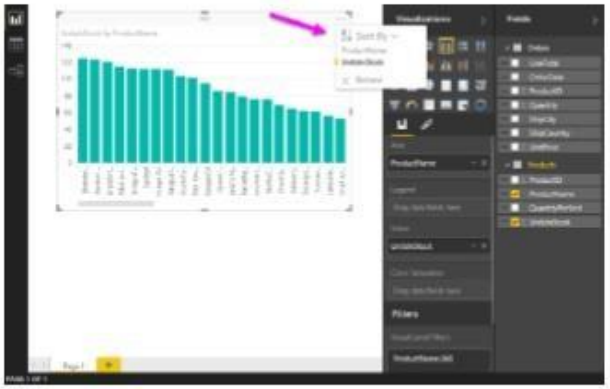


6. Select Cancel, and then select Relationship view in Power BI Desktop.



Task 4: Build visuals using your data

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



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 theValues field; the circles on the map for each country are now relative in size to the LineTotal for orders shipped to that country.