DATA SCIENCE

PRACTICAL 1

A) CSV TO HOURS

CODE:

```
# Utility Start CSV to HORUS =====================================
# Standard Tools
import pandas as pd
# Input Agreement =================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values =======================================')
print(InputData)
print('===========================================================')
# Processing Rules ================================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =====================================')
print(ProcessData)
print('============================================================')
# Output Agreement ================================================
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
# Utility done =====================================================
```

:

B) XML TO HOURS

CODE:

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

print('=====================================================')
print(InputData)
print('=====================================================')
# Processing Rules ===========================================
ProcessDataXML=InputData
# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
```

```
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('=====================================================')
print('Process Data Values =================================')
print('=====================================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ==========================================
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('XML to HORUS - Done')
print('=====================================================')
# Utility done ==============================================
```

JSON TO HOURS

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

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

C) MYSQL DATABASE TO HOURS

CODE:

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

D) PICTURE (JPEG) TO HOURS

CODE:

```
import sys
import os
#from scipy.misc import imread
from skimage import io
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
# Input Agreement =============================================
sInputFileName=Base + '/05-DS/9999-Data/Angus.jpg'
InputData = io.imread(sInputFileName, as_gray=False, pilmode='RGBA')
print('Input Data Values ====================================')
print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])
print('======================================================')
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
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=Base + '/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('======================================================')
print('Picture to HORUS - Done')
print('======================================================')
# Utility done ================================================
```

E) VIDEO TO HOURS

CODE:

MOVIE TO FRAMES

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

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

FRAMES TO HOURS:
CODE:

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

```
    OutputData=ProcessData
    print('Storing File')
    sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Movie-Frame.csv'
    OutputData.to_csv(sOutputFileName, index = False)
    print('=====================================================')
    print('Processed ; ', f,' frames')
    print('=====================================================')
    print('Movie to HORUS - Done')
    print('=====================================================')
```

F) AUDIO TO HOURS

CODE:

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

```
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('====================================================')
print('Processing : ', sInputFileName)
print('====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("4 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('====================================================')
print('Processing : ', sInputFileName)
print('====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("6 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/8ch-sound.wav'
print('====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("8 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('====================================================')
print('Audio to HORUS - Done')
print('=====================================================')
#============================================================
# Utility done =============================================
#============================================================
```

PRACTICAL 2 (In Python)

A) UTILITIES AND AUDITING

CODE:

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

B) Data Binning and Bucketing

CODE:

```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import scipy.stats as stats
np.random.seed(0)
# example data
mu = 90 # mean of distribution
sigma = 25 # standard deviation of distribution
x = mu + sigma * np.random.randn(5000)
num_bins = 25
fig, ax = plt.subplots()
# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, density=1)
# add a 'best fit' line
y = stats.norm.pdf(bins, mu, sigma)
# mlab.normpdf(bins, mu, sigma)
ax.plot(bins, y, '--')
ax.set_xlabel('Example Data')
ax.set_ylabel('Probability density')
sTitle=r'Histogram ' + str(len(x)) + ' entries into ' + str(num_bins) + ' Bins: $\mu=' + str(mu) + '$,
$\sigma=' + str(sigma) + '$'
ax.set_title(sTitle)
fig.tight_layout()
sPathFig='C:/VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png'
fig.savefig(sPathFig)
plt.show()
```

C**. Averaging Of Data**
CODE:

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

D. **Outlier Detection**
CODE:

```
import pandas as pd
InputFileName='IP_DATA_CORE.csv'
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=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
```

```
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) & (AllData.Latitude<=UpperBound)]
print(OutliersNot)
```

Audit
CODE:

```
import sys
import os
import logging
import uuid
import shutil
import time
############################################################
Base='C:/VKHCG'
############################################################
sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04-Clark']
sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-Organise','06-Report']
sLevels=['debug','info','warning','error']
for sCompany in sCompanies:
    sFileDir=Base + '/' + sCompany
    if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
    for sLayer in sLayers:
        log = logging.getLogger() # root logger
    for hdlr in log.handlers[:]: # remove all old handlers
        log.removeHandler(hdlr)
#------------------------------------------------------------------------------------------------------------
    sFileDir=Base + '/' + sCompany + '/' + sLayer + '/Logging'
    if os.path.exists(sFileDir):
        shutil.rmtree(sFileDir)
    time.sleep(2)
    if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
skey=str(uuid.uuid4())
sLogFile=Base + '/' + sCompany + '/' + sLayer + '/Logging/Logging_'+skey+'.log'
print('Set up:',sLogFile)
# set up logging to file - see previous section for more details
logging.basicConfig(level=logging.DEBUG,
    format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
    datefmt='%m-%d %H:%M',
    filename=sLogFile,
    filemode='w')
# define a Handler which writes INFO messages or higher to the sys.stderr
console = logging.StreamHandler()
console.setLevel(logging.INFO)
# set a format which is simpler for console use
formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
# tell the handler to use this format
console.setFormatter(formatter)
# add the handler to the root logger
logging.getLogger('').addHandler(console)
# Now, we can log to the root logger, or any other logger. First the root...
logging.info('Practical Data Science is fun!.')
for sLevel in sLevels:
    sApp='Apllication-'+ sCompany + '-' + sLayer + '-' + sLevel
logger = logging.getLogger(sApp)
```

```
if sLevel == 'debug':
    logger.debug('Practical Data Science logged a debugging message.')
if sLevel == 'info':
    logger.info('Practical Data Science logged information message.')
if sLevel == 'warning':
    logger.warning('Practical Data Science logged a warning message.')
if sLevel == 'error':
    logger.error('Practical Data Science logged an error message.')
```

PRACTICAL 3

A Retrieve Superstep

In R studio

```
library(readr)
IP_DATA_ALL <- read_csv("C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
  cols(
    ID = col_double(),
    Country = col_character(),
    `Place Name` = col_character(),
    `Post Code` = col_double(),
    Latitude = col_double(),
    Longitude = col_double(),
    `First IP Number` = col_double(),
    `Last IP Number` = col_double()
  );
View(IP_DATA_ALL)
spec(IP_DATA_ALL)
cols(
  ID = col_double(),
  Country = col_character(),
  `Place Name` = col_character(),
  `Post Code` = col_double(),
  Latitude = col_double(),
  Longitude = col_double(),
  `First IP Number` = col_double(),
  `Last IP Number` = col_double()
);
library(tibble)
set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)
Place Name -> Place.Name
Post Code -> Post.Code
First IP Number -> First.IP.Number
Last IP Number -> Last.IP.Number
sapply(IP_DATA_ALL_FIX, typeof)
library(data.table)
hist_country=data.table(Country=unique(IP_DATA_ALL_FIX[is.na(IP_DATA_ALL_FIX
['Country'])==0,]$Country))
setorder(hist_country,'Country')
hist_country_with_id=rowid_to_column(hist_country, var = "RowIDCountry")
View(hist_country_fix)
```

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

B: **Program to retrieve different attributes of data**

```
import sys
import os
import pandas as pd
######################################################################
Base='C:/VKHCG'
######################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
######################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
print('Rows:', IP_DATA_ALL.shape[0])
print('Columns:', IP_DATA_ALL.shape[1])
print('### Raw Data Set ########################################')
for i in range(0,len(IP_DATA_ALL.columns)):

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

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

C. DATA PATTERN

To determine a pattern of the data values, Replace all alphabet values with an uppercase case *A*, all numbers with an uppercase *N*, and replace any spaces with a lowercase letter *b* and all other unknown characters with a lowercase *u*. As a result, “Good Book 101” becomes AAAAbAAAAbNNNu”. This pattern creation is beneficial for designing any specific assess rules. This pattern view of data is a quick way to identify common patterns or determine standard layouts.

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

Example 2: This is a common use of patterns to separate common standards and structures. Pattern can be loaded in separate retrieve procedures. If the same two patterns, NNNNuNNuNN and uuNNuNNuNN, are found, you can send NNNNuNNuNN directly to be converted into a date, while uuNNuNNuNN goes through a quality-improvement process to then route back to the same queue as NNNNuNNuNN, once it complies.

```
library(readr)
library(data.table)
Base='C:/VKHCG'
FileName=paste0(Base,'/01-Vermeulen/00-RawData/IP_DATA_ALL.csv')
IP_DATA_ALL <- read_csv(FileName)
hist_latitude=data.table(Latitude=unique(IP_DATA_ALL$Latitude))
pattern_latitude=data.table(latitude=hist_latitude$Latitude,
                            Patternlatitude=as.character(hist_latitude$Latitude))
oldchar=c(letters,LETTERS)
newchar=replicate(length(oldchar),"A")
for (r in seq(nrow(pattern_latitude))){
s=pattern_latitude[r,]$Patternlatitude;
for (c in seq(length(oldchar))){
    s=chartr(oldchar[c],newchar[c],s)
  };
  for (n in seq(0,9,1)){
    s=chartr(as.character(n),"N",s)
  };
  s=chartr(" ","b",s)
  s=chartr("+","u",s)
  s=chartr("-","u",s)
  s=chartr(".","u",s)
  pattern_latitude[r,]$Patternlatitude=s;
};
setorder(pattern_latitude,latitude)
View(pattern_latitude[1:3])
```

D.**Loading IP_DATA_ALL:**

```
import sys
import os
import pandas as pd
Base='C:/VKHCG'
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
print('Rows:', IP_DATA_ALL.shape[0])
print('Columns:', IP_DATA_ALL.shape[1])
print('### Raw Data Set #######################################')
for i in range(0,len(IP_DATA_ALL.columns)):
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print('### Fixed Data Set #####################################')
IP_DATA_ALL_FIX=IP_DATA_ALL
for i in range(0,len(IP_DATA_ALL.columns)):
    cNameOld=IP_DATA_ALL_FIX.columns[i] + ' '
cNameNew=cNameOld.strip().replace(" ", ".")
IP_DATA_ALL_FIX.columns.values[i] = cNameNew
print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print(IP_DATA_ALL_FIX.head())
print('Fixed Data Set with ID')
IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
IP_DATA_ALL_with_ID.index.names = ['RowID']
#print(IP_DATA_ALL_with_ID.head())
sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")
print('### Done!! ##############################################')
```

PRACTICAL 4
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**

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/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('###################################')
print('## Raw Data Values')
print('###################################')
print(RawData)
print('###################################')
print('## Data Profile')
print('###################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('###################################')
######################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
TestData=RawData.dropna(axis=1, how='all')
```

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

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

```
20 10.0 A B C 2.0 111.0 21
##################################
## Data Profile
##################################
Rows : 21
Columns : 7
##################################
##################################
### Done!! ######################
##################################
>>>
```

All of column E has been deleted, owing to the fact that all values in that column were missing values/errors.

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

```
################## Assess-Good-Bad-02.py#############################
# -*- coding: utf-8 -*-
####################################################################
import sys
import os
import pandas as pd
####################################################################
Base='C:/VKHCG'
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-02.csv'
Company='01-Vermeulen'
####################################################################
Base='C:/VKHCG'
####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
####################################################################
### Import Warehouse
####################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('##################################')
print('## Raw Data Values')
print('##################################')
print(RawData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('#####################################')
###########################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
###########################################################################
TestData=RawData.dropna(axis=1, how='any')
###########################################################################
print('#####################################')
print('## Test Data Values')
print('#####################################')
print(TestData)
print('#####################################')
print('## Data Profile')
print('#####################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('#####################################')
###########################################################################
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
###########################################################################
print('#####################################')
print('### Done!! ########################')
print('#####################################')
###########################################################################
```

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

```
########### Assess-Network-Routing-Company.py #######################
import sys
import os
import pandas as pd
###########################################################################
pd.options.mode.chained_assignment = None
###########################################################################
Base='C:/VKHCG'
###########################################################################
print('#####################################')
print('Working Base :',Base, ' using Windows')
print('#####################################')
###########################################################################
sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sInputFileName3='01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv'
###########################################################################
```

```
sOutputFileName='Assess-Network-Routing-Company.csv'
Company='01-Vermeulen'
##################################################################
##################################################################
### Import Country Data
##################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('################################')
print('Loading :',sFileName)
print('################################')
CountryData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
print('################################')
##################################################################
## Assess Country Data
##################################################################
print('################################')
print('Changed :',CountryData.columns.values)
CountryData.rename(columns={'Country': 'Country_Name'}, inplace=True)
CountryData.rename(columns={'ISO-2-CODE': 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inplace=True)
CountryData.drop('ISO-3-Code', axis=1, inplace=True)
CountryData.drop('RowID', axis=1, inplace=True)
print('To :',CountryData.columns.values)
print('################################')
##################################################################
##################################################################
### Import Company Data
##################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('################################')
##################################################################
## Assess Company Data
##################################################################
print('################################')
print('Changed :',CompanyData.columns.values)
CompanyData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CompanyData.columns.values)
print('################################')
##################################################################
##################################################################
### Import Customer Data
##################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName3
print('################################')
print('Loading :',sFileName)
print('################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('################################')
print('Loaded Customer :',CustomerRawData.columns.values)
print('################################')
################################################################
CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('################################')
print('Remove Blank Country Code')
print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])
print('################################')
################################################################
print('################################')
print('Changed :',CustomerData.columns.values)
CustomerData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CustomerData.columns.values)
print('################################')
################################################################
print('################################')
print('Merge Company and Country Data')
print('################################')
CompanyNetworkData=pd.merge(CompanyData,
CountryData,
how='inner',
on='Country_Code'
)
################################################################
print('################################')
print('Change ',CompanyNetworkData.columns.values)
for i in CompanyNetworkData.columns.values:
j='Company_'+i
CompanyNetworkData.rename(columns={i: j}, inplace=True)
print('To ', CompanyNetworkData.columns.values)
print('################################')
################################################################
################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
################################################################
sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
CompanyNetworkData.to_csv(sFileName, index = False, encoding="latin-1")
################################################################
################################################################
print('################################')
print('### Done!! #####################')
print('################################')
################################################################
```

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

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

C. Write a Python / R program to build directed acyclic graph.

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

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

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

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

```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName='Assess-DAG-Company-GPS.png'
Company='01-Vermeulen'
### Import Company Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('################################')
print(CompanyData)
print('################################')
print('Rows : ',CompanyData.shape[0])
print('################################')
G=nx.Graph()
for i in range(CompanyData.shape[0]):
nLatitude=round(CompanyData['Latitude'][i],2)
nLongitude=round(CompanyData['Longitude'][i],2)
if nLatitude < 0:
sLatitude = str(nLatitude*-1) + ' S'
else:
sLatitude = str(nLatitude) + ' N'
if nLongitude < 0:
sLongitude = str(nLongitude*-1) + ' W'
else:
sLongitude = str(nLongitude) + ' E'
sGPS= sLatitude + '-' + sLongitude
G.add_node(sGPS)
print('################################')
for n1 in G.nodes():
for n2 in G.nodes():
if n1 != n2:
print('Link :',n1,' to ', n2)
G.add_edge(n1,n2)
print('################################')
print('################################')
print("Nodes of graph: ")
print(G.number_of_nodes())
```

D. Write a Python / R program to build directed acyclic graph.

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

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

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

```
plt.savefig(sFileName) # save as png
plt.show()
```

Practical 5

Processing Data

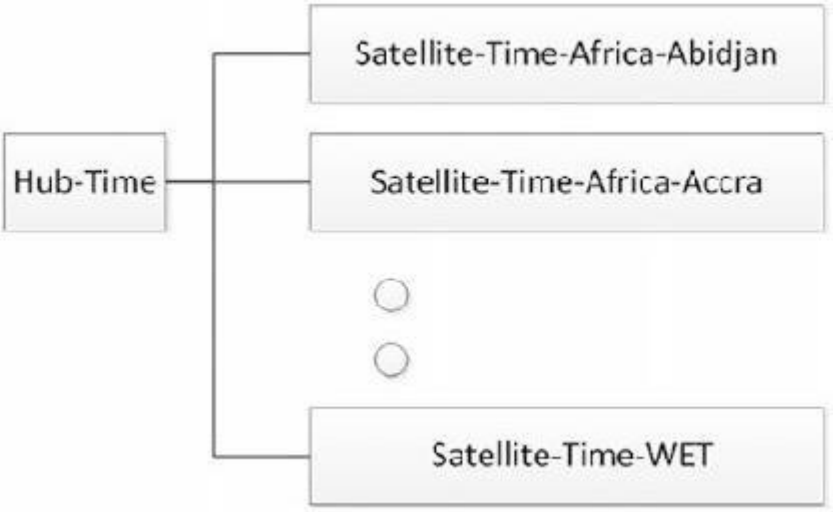
A. Build the time hub, links, and satellites.

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

```
####################################################################
# -*- coding: utf-8 -*-
####################################################################
import sys
import os
from datetime import datetime
from datetime import timedelta
from pytz import timezone, all_timezones
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
pd.options.mode.chained_assignment = None
####################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
####################################################################
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
####################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
####################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
####################################################################
base = datetime(2018,1,1,0,0,0)
numUnits=10*365*24
####################################################################
```

```
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)]
t=0
for i in date_list:
now_utc=i.replace(tzinfo=timezone('UTC'))
sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S")
print(sDateTime)
sDateTimeKey=sDateTime.replace(' ','-').replace(':','-')
t+=1
IDNumber=str(uuid.uuid4())
TimeLine=[('ZoneBaseKey', ['UTC']),
('IDNumber', [IDNumber]),
('nDateTimeValue', [now_utc]),
('DateTimeValue', [sDateTime]),
('DateTimeKey', [sDateTimeKey])]
if t==1:
TimeFrame = pd.DataFrame.from_items(TimeLine)
else:
TimeRow = pd.DataFrame.from_items(TimeLine)
TimeFrame = TimeFrame.append(TimeRow)
####################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
####################################################################
TimeFrame.set_index(['IDNumber'],inplace=True)
####################################################################
sTable = 'Process-Time'
print('Storing :',sDatabaseName,' Table:',sTable)
TimeHubIndex.to_sql(sTable, conn1, if_exists="replace")
####################################################################
sTable = 'Hub-Time'
print('Storing :',sDatabaseName,' Table:',sTable)
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
####################################################################
active_timezones=all_timezones
z=0
for zone in active_timezones:
t=0
for j in range(TimeFrame.shape[0]):
now_date=TimeFrame['nDateTimeValue'][j]
DateTimeKey=TimeFrame['DateTimeKey'][j]
now_utc=now_date.replace(tzinfo=timezone('UTC'))
sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S")
now_zone = now_utc.astimezone(timezone(zone))
sZoneDateTime=now_zone.strftime("%Y-%m-%d %H:%M:%S")
print(sZoneDateTime)
t+=1
z+=1
IDZoneNumber=str(uuid.uuid4())
TimeZoneLine=[('ZoneBaseKey', ['UTC']),
('IDZoneNumber', [IDZoneNumber]),
('DateTimeKey', [DateTimeKey]),
('UTCDateTimeValue', [sDateTime]),
('Zone', [zone]),
('DateTimeValue', [sZoneDateTime])]
if t==1:
TimeZoneFrame = pd.DataFrame.from_items(TimeZoneLine)
else:
TimeZoneRow = pd.DataFrame.from_items(TimeZoneLine)
TimeZoneFrame = TimeZoneFrame.append(TimeZoneRow)
TimeZoneFrameIndex=TimeZoneFrame.set_index(['IDZoneNumber'],inplace=False)
sZone=zone.replace('/','-').replace(' ','')
#############################################################
sTable = 'Process-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn1, if_exists="replace")
#################################################################
#############################################################
sTable = 'Satellite-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('################')
#################################################################
print('### Done!! ############################################')
#################################################################
```

You have built your first hub and satellites for time in the data vault.
The data vault has been built in directory ..\ VKHCG\88-DV\datavault.db. You can access it with your SQLite
Tools



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

nominal.
The proram will generate a golden nominal using our sample data set.
Open your Python editor and create a file called Process-People.py in the ..
C:\VKHCG\04-Clark\03-Process directory.

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

import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
from datetime import datetime, timedelta
from pytz import timezone, all_timezones
from random import randint
import uuid
################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='04-Clark'
sInputFileName='02-Assess/01-EDS/02-Python/Assess_People.csv'
################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
################################################################
### Import Female Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
print(sFileName)
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)
start_date = datetime(1900,1,1,0,0,0)
```

```
start_date_utc=start_date.replace(tzinfo=timezone('UTC'))
HoursBirth=100*365*24
RawData['BirthDateUTC']=RawData.apply(lambda row:
(start_date_utc + timedelta(hours=randint(0, HoursBirth)))
,axis=1)
zonemax=len(all_timezones)-1
RawData['TimeZone']=RawData.apply(lambda row:
(all_timezones[randint(0, zonemax)])
,axis=1)
RawData['BirthDateISO']=RawData.apply(lambda row:
row["BirthDateUTC"].astimezone(timezone(row['TimeZone']))
,axis=1)
RawData['BirthDateKey']=RawData.apply(lambda row:
row["BirthDateUTC"].strftime("%Y-%m-%d %H:%M:%S")
,axis=1)
RawData['BirthDate']=RawData.apply(lambda row:
row["BirthDateISO"].strftime("%Y-%m-%d %H:%M:%S")
,axis=1)
RawData['PersonID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1)
################################################################
Data=RawData.copy()
Data.drop('BirthDateUTC', axis=1,inplace=True)
Data.drop('BirthDateISO', axis=1,inplace=True)
indexed_data = Data.set_index(['PersonID'])
print('################################')
#################################################################
print('################')
sTable='Process_Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_data.to_sql(sTable, conn1, if_exists="replace")
print('################')
################################################################
PersonHubRaw=Data[['PersonID','FirstName','SecondName','LastName','BirthDateKey']]
PersonHubRaw['PersonHubID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1)
PersonHub=PersonHubRaw.drop_duplicates(subset=None, \
keep='first',\
inplace=False)
indexed_PersonHub = PersonHub.set_index(['PersonHubID'])
sTable = 'Hub-Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonHub.to_sql(sTable, conn2, if_exists="replace")
################################################################
PersonSatelliteGenderRaw=Data[['PersonID','FirstName','SecondName','LastName'\
,'BirthDateKey','Gender']]
PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1)
```

```
PersonSatelliteGender=PersonSatelliteGenderRaw.drop_duplicates(subset=None, \
keep='first', \
inplace=False)
indexed_PersonSatelliteGender = PersonSatelliteGender.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Gender'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteGender.to_sql(sTable, conn2, if_exists="replace")
########################################################################
PersonSatelliteBirthdayRaw=Data[['PersonID','FirstName','SecondName','LastName',\
'BirthDateKey','TimeZone','BirthDate']]
PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1)
PersonSatelliteBirthday=PersonSatelliteBirthdayRaw.drop_duplicates(subset=None, \
keep='first',\
inplace=False)
indexed_PersonSatelliteBirthday = PersonSatelliteBirthday.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Names'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteBirthday.to_sql(sTable, conn2, if_exists="replace")
########################################################################
sFileDir=Base + '/' + Company + '/03-Process/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
########################################################################
sOutputFileName = sTable + '.csv'
sFileName=sFileDir + '/' + sOutputFileName
print('####################################')
print('Storing :', sFileName)
print('####################################')
RawData.to_csv(sFileName, index = False)
print('####################################')
#########################################################################
print('################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('################')
#########################################################################
print('### Done!! ##################################################')
#########################################################################
```

Output :

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

Load the person into the data vault

```
========== RESTART: C:\VKHCG\04-Clark\03-Process\Process-People.py ==========
####################################
```

```
Working Base : C:/VKHCG using win32
##################################
##################################
Loading : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
##################################
C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
##################################
#################
Storing : C:/VKHCG/88-DV/datavault.db Table: Process_Person
#################
Storing : C:/VKHCG/88-DV/datavault.db Table: Satellite-Person-Gender
Storing : C:/VKHCG/88-DV/datavault.db Table: Satellite-Person-Names
##################################
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Satellite-Person-Names.csv
##################################
##################################
#################
Vacuum Databases
#################
### Done!! ###############################################
```

Vehicles

The international classification of vehicles is a complex process. There are standards, but these are not universally applied or similar between groups or countries.

Let’s load the vehicle data for Hillman Ltd into the data vault, as we will need it later. Create a new file named
Process-Vehicle-Logistics.py in the Python editor in directory ..\VKHCG\03-Hillman\03-Process.

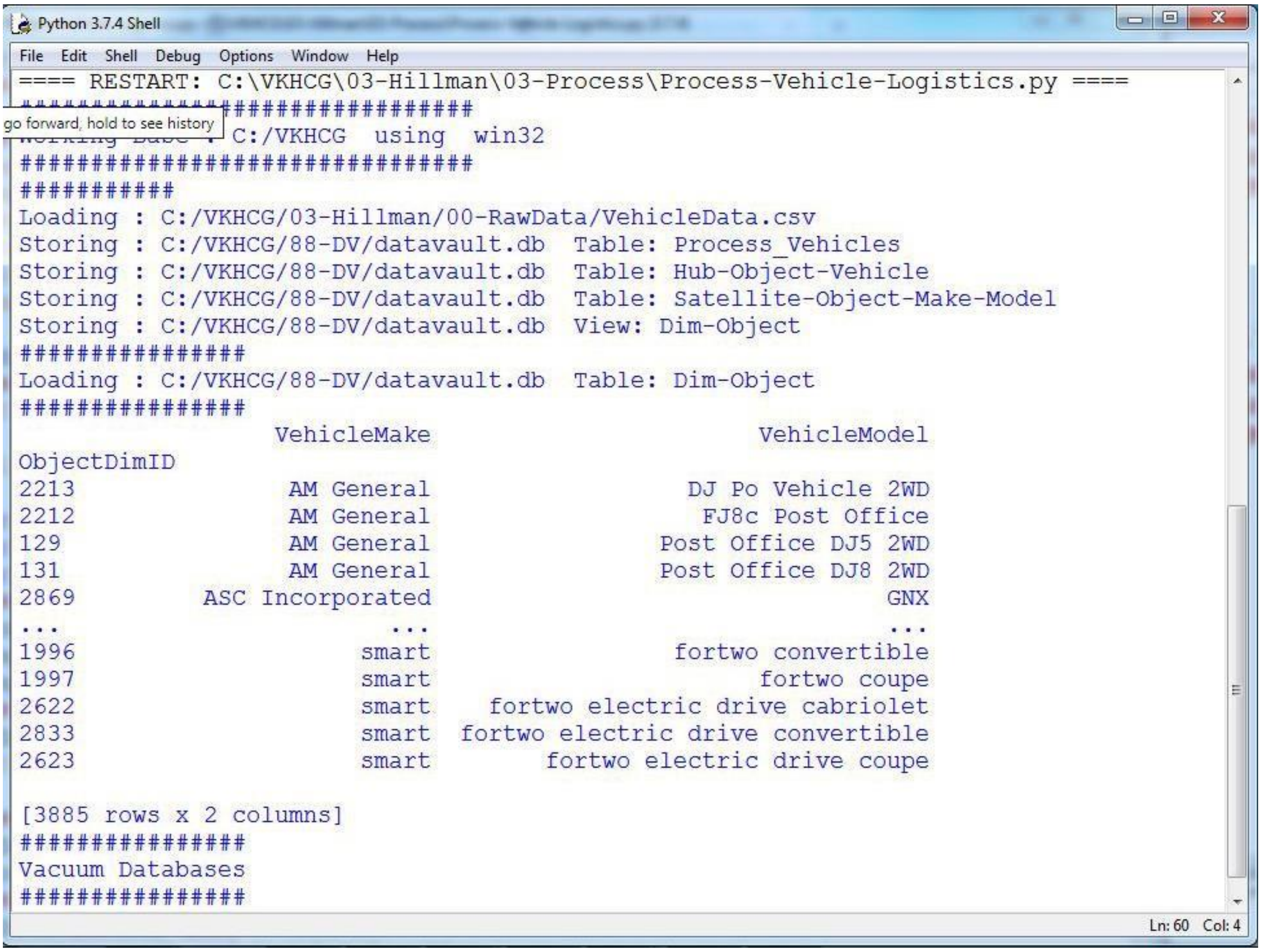
```
####################################################################
# -*- coding: utf-8 -*-
####################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
pd.options.mode.chained_assignment = None
####################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
####################################################################
Company='03-Hillman'
InputDir='00-RawData'
```

```
InputFileName='VehicleData.csv'
####################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
####################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
####################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('###########')
print('Loading :',sFileName)
VehicleRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
####################################################################
sTable='Process_Vehicles'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleRaw.to_sql(sTable, conn1, if_exists="replace")
####################################################################
VehicleRawKey=VehicleRaw[['Make','Model']].copy()
VehicleKey=VehicleRawKey.drop_duplicates()
####################################################################
VehicleKey['ObjectKey']=VehicleKey.apply(lambda row:
str('('+ str(row['Make']).strip().replace(' ', '-').replace('/', '-').lower() +
')-(' + (str(row['Model']).strip().replace(' ', '-').replace(' ', '-').lower())
+')')
,axis=1)
####################################################################
VehicleKey['ObjectType']=VehicleKey.apply(lambda row:
'vehicle'
,axis=1)
####################################################################
VehicleKey['ObjectUUID']=VehicleKey.apply(lambda row:
str(uuid.uuid4())
,axis=1)
####################################################################
### Vehicle Hub
####################################################################
#
VehicleHub=VehicleKey[['ObjectType','ObjectKey','ObjectUUID']].copy()
VehicleHub.index.name='ObjectHubID'
sTable = 'Hub-Object-Vehicle'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleHub.to_sql(sTable, conn2, if_exists="replace")
####################################################################
```

```
### Vehicle Satellite
#################################################################
#
VehicleSatellite=VehicleKey[['ObjectType','ObjectKey','ObjectUUID','Make','Model']].copy()
VehicleSatellite.index.name='ObjectSatelliteID'
sTable = 'Satellite-Object-Make-Model'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleSatellite.to_sql(sTable, conn2, if_exists="replace")
#################################################################
### Vehicle Dimension
#################################################################
sView='Dim-Object'
print('Storing :',sDatabaseName,' View:',sView)
sSQL="CREATE VIEW IF NOT EXISTS [" + sView + "] AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " H.ObjectType,"
sSQL=sSQL+ " H.ObjectKey AS VehicleKey,"
sSQL=sSQL+ " TRIM(S.Make) AS VehicleMake,"
sSQL=sSQL+ " TRIM(S.Model) AS VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " [Hub-Object-Vehicle] AS H"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " [Satellite-Object-Make-Model] AS S"
sSQL=sSQL+ " ON"
sSQL=sSQL+ " H.ObjectType=S.ObjectType"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " H.ObjectUUID=S.ObjectUUID;"
sql.execute(sSQL,conn2)
print('################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " VehicleMake,"
sSQL=sSQL+ " VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " [" + sView + "]"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " VehicleMake"
sSQL=sSQL+ " AND"
DimObjectData=pd.read_sql_query(sSQL, conn2)
DimObjectData.index.name='ObjectDimID'
DimObjectData.sort_values(['VehicleMake','VehicleModel'],inplace=True, ascending=True)
print('################')
print(DimObjectData)
##################################################################
print('################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('################')
##################################################################
```

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

Output:


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
go forward, hold to see history
Ln: 60 Col: 4

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

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

Human-Environment Interaction

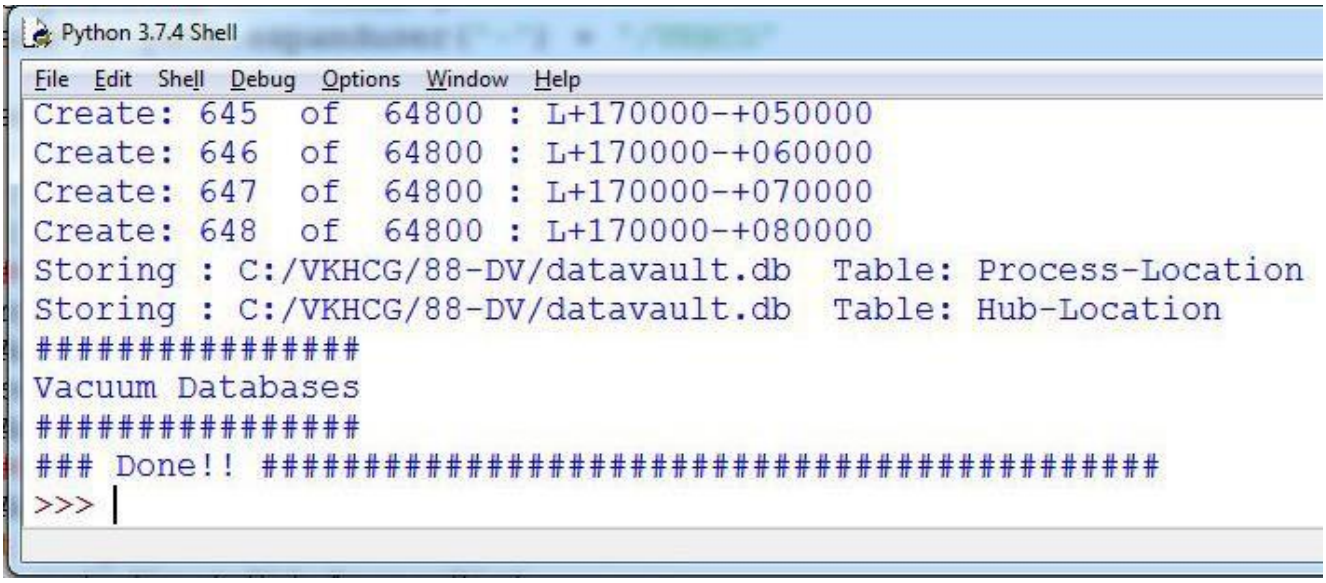
The interaction of humans with their environment is a major relationship that guides people's behavior and the characteristics of the location. Activities such as mining and other industries, roads, and landscaping at a location create both positive and negative effects on the environment, but also on humans. A location earmarked as a green belt, to assist in reducing the carbon footprint, or a new interstate change its current and future characteristics. The location is a main data source for the data science, and, normally, we find unknown or unexpected effects on the data insights. In the Python editor, open a new file named Process_Location.py in directory ..\VKHCG\01-Vermeulen\03-Process.

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

```
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='01-Vermeulen'
InputAssessGraphName='Assess_All_Animals.gml'
EDSAssessDir='02-Assess/01-EDS'
InputAssessDir=EDSAssessDir + '/02-Python'
################################################################
sFileAssessDir=Base + '/' + Company + '/' + InputAssessDir
if not os.path.exists(sFileAssessDir):
os.makedirs(sFileAssessDir)
################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
t=0
tMax=360*180
################################################################
for Longitude in range(-180,180,10):
for Latitude in range(-90,90,10):
t+=1
IDNumber=str(uuid.uuid4())
LocationName='L'+format(round(Longitude,3)*1000, '+07d') +\
'-'+format(round(Latitude,3)*1000, '+07d')
print('Create:',t,' of ',tMax,':',LocationName)
LocationLine=[('ObjectBaseKey', ['GPS']),
('IDNumber', [IDNumber]),
('LocationNumber', [str(t)]),
('LocationName', [LocationName]),
('Longitude', [Longitude]),
('Latitude', [Latitude])]
if t==1:
```

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

Output:



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

Forecasting

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

- What should we buy?

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

People want to know what you calculate to determine what is about to happen.
Open a new file in your Python editor and save it as Process-Shares-Data.py in directory C: \VKHCG\04-Clark\03-Process. I will guide you through this process. You will require a library called quandl

type pip install quandl in cmd

```
######################################################################
import sys
import os
import sqlite3 as sq
import quandl
import pandas as pd
######################################################################
Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
######################################################################
Company='04-Clark'
sInputFileName='00-RawData/VKHCG_Shares.csv'
sOutputFileName='Shares.csv'
######################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
######################################################################
sFileDir1=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir1):
os.makedirs(sFileDir1)
######################################################################
sFileDir2=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir2):
os.makedirs(sFileDir2)
######################################################################
sFileDir3=Base + '/' + Company + '/03-Process/01-EDS/02-Python'
if not os.path.exists(sFileDir3):
os.makedirs(sFileDir3)
######################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
######################################################################
### Import Share Names Data
######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Rows :',RawData.shape[0])
print('Columns:',RawData.shape[1])
print('################')
################################################################
sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
################################################################
sFileName=sFileDir2 + '/Assess_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
################################################################
sFileName=sFileDir3 + '/Process_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
################################################################
### Import Shares Data Details
nShares=RawData.shape[0]
#nShares=6
for sShare in range(nShares):
sShareName=str(RawData['Shares'][sShare])
ShareData = quandl.get(sShareName)
UnitsOwn=RawData['Units'][sShare]
ShareData['UnitsOwn']=ShareData.apply(lambda row:(UnitsOwn),axis=1)
ShareData['ShareCode']=ShareData.apply(lambda row:(sShareName),axis=1)
print('################')
print('Share :',sShareName)
print('Rows :',ShareData.shape[0])
print('Columns:',ShareData.shape[1])
print('################')
#################################################################
print('################')
sTable=str(RawData['sTable'][sShare])
print('Storing :',sDatabaseName,' Table:',sTable)
ShareData.to_sql(sTable, conn, if_exists="replace")
print('################')
################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
ShareData.to_csv(sFileName, index = False)
```

```
print('###################################')
######################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir2 + '/Assess_' + sOutputFileName
print('###################################')
print('Storing :', sFileName)
print('###################################')
ShareData.to_csv(sFileName, index = False)
print('###################################')
######################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir3 + '/Process_' + sOutputFileName
print('###################################')
print('Storing :', sFileName)
print('###################################')
ShareData.to_csv(sFileName, index = False)
print('###################################')
print('### Done!! ################################################')
######################################################################
```

Output:

```
======== RESTART: C:\VKHCG\04-Clark\03-Process\Process-Shares-Data.py ========
Working Base : C:/VKHCG using win32
Loading : C:/VKHCG/04-Clark/00-RawData/VKHCG_Shares.csv
Rows : 10
Columns: 3
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_Shares.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_Shares.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_Shares.csv
Share : WIKI/GOOGL
Rows : 3424
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Google
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Google.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Google.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Google.csv
Share : WIKI/MSFT
Rows : 8076
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Microsoft
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Microsoft.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Microsoft.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Microsoft.csv
Share : WIKI/UPS
Rows : 4622
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_UPS
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_UPS.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_UPS.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_UPS.csv
Share : WIKI/AMZN
```

```
Rows : 5248
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Amazon
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Amazon.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Amazon.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Amazon.csv
Share : LOCALBTC/USD
Rows : 1863
Columns: 6
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: LOCALBTC_USD
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_LOCALBTC_USD.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_LOCALBTC_USD.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_LOCALBTC_USD.csv
Share : PERTH/AUD_USD_M
Rows : 340
Columns: 8
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: PERTH_AUD_USD_M
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_PERTH_AUD_USD_M.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_PERTH_AUD_USD_M.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_PERTH_AUD_USD_M.csv
Share : PERTH/AUD_USD_D
Rows : 7989
Columns: 8
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: PERTH_AUD_USD_D
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_PERTH_AUD_USD_D.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_PERTH_AUD_USD_D.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_PERTH_AUD_USD_D.csv
Share : FRED/GDP
Rows : 290
Columns: 3
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: FRED/GDP
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_FRED-GDP.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_FRED-GDP.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_FRED-GDP.csv
Share : FED/RXI_US_N_A_UK
Rows : 49
Columns: 3
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: FED_RXI_US_N_A_UK
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_FED_RXI_US_N_A_UK.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_FED_RXI_US_N_A_UK.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_FED_RXI_US_N_A_UK.csv
Share : FED/RXI_N_A_CA
Rows : 49
Columns: 3
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: FED_RXI_N_A_CA
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_FED_RXI_N_A_CA.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_FED_RXI_N_A_CA.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_FED_RXI_N_A_CA.csv
### Done!! ############################################
```

Practical 6
Transforming Data

Transform Superstep

The Transform superstep allows you, as a data scientist, to take data from the data vault and formulate answers to questions raised by your investigations. The transformation step is the data science process that converts results into insights. It takes standard data science techniques and methods to attain insight and knowledge about the data that then can be transformed into actionable decisions, which, through storytelling, you can explain to non-data scientists what you have discovered in the data lake.

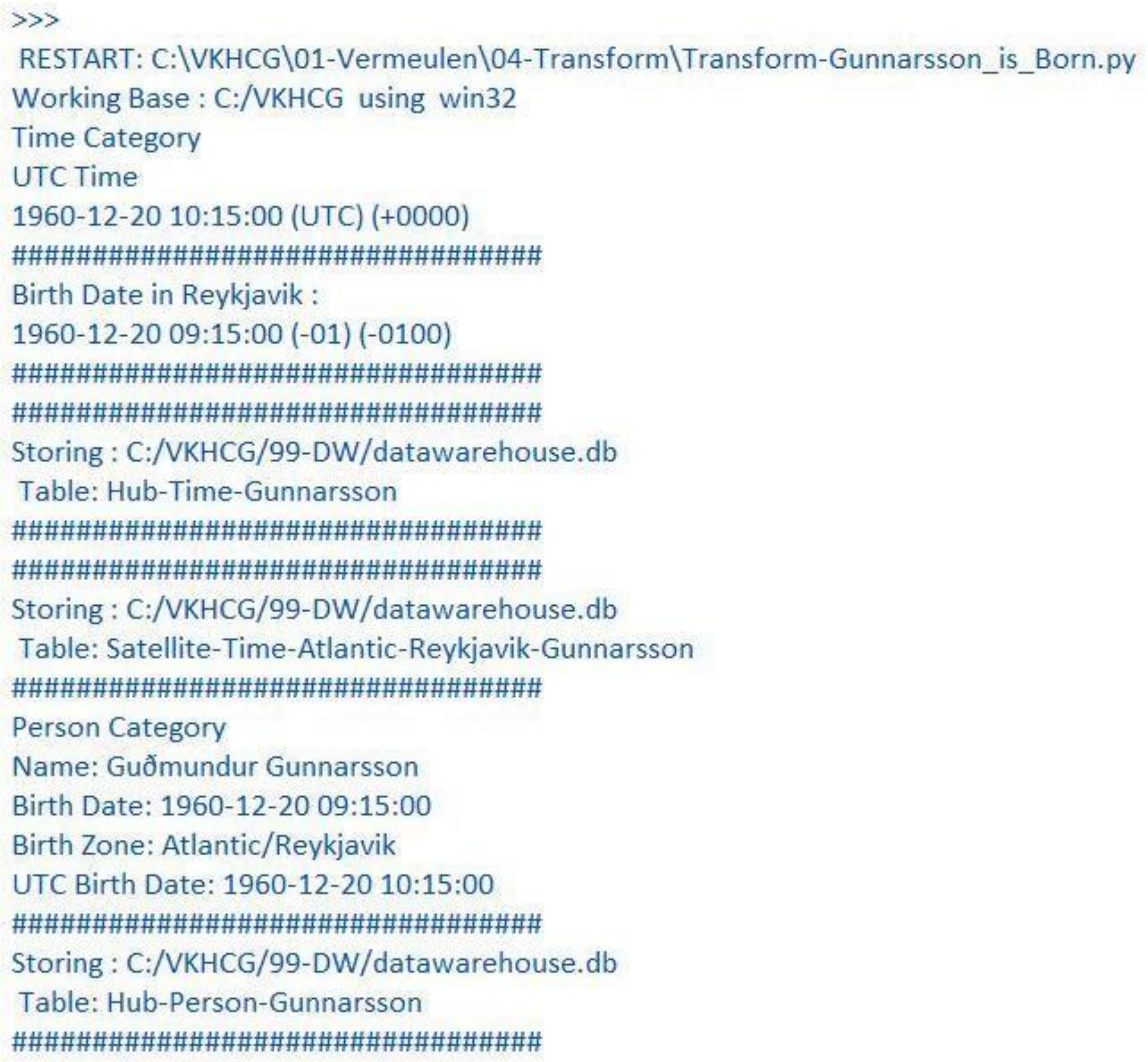
To illustrate the consolidation process, the example show a person being borne. Open a new file in the Python editor and save it as Transform-Gunnarsson_is_Born.py in directory C: \VKHCG\01-Vermeulen\04-Transform.

```
####################################################################
# -*- coding: utf-8 -*-
####################################################################
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
####################################################################
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
####################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
os.makedirs(sDataVaultDir)
####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
```

```
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
################################################################
print('\n#################################')
print('Time Category')
print('UTC Time')
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
print(BirthDateZoneUTCStr)
print('#################################')
print('Birth Date in Reykjavik :')
BirthZone = 'Atlantic/Reykjavik'
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
print(BirthDateStr)
print('#################################')
################################################################
IDZoneNumber=str(uuid.uuid4())
sDateTimeKey=BirthDateZoneStr.replace(' ','-').replace(':','-')
TimeLine=[('ZoneBaseKey', ['UTC']),
('IDNumber', [IDZoneNumber]),
('DateTimeKey', [sDateTimeKey]),
('UTCDateTimeValue', [BirthDateZoneUTC]),
('Zone', [BirthZone]),
('DateTimeValue', [BirthDateStr])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
################################################################
sTable = 'Hub-Time-Gunnarsson'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Time-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
################################################################
TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']]
TimeSatelliteIndex=TimeSatellite.set_index(['IDNumber'],inplace=False)
################################################################
BirthZoneFix=BirthZone.replace(' ','-').replace('/','-')
sTable = 'Satellite-Time-' + BirthZoneFix + '-Gunnarsson'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeSatelliteIndex.to_sql(sTable, conn2, if_exists="replace")
```

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

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



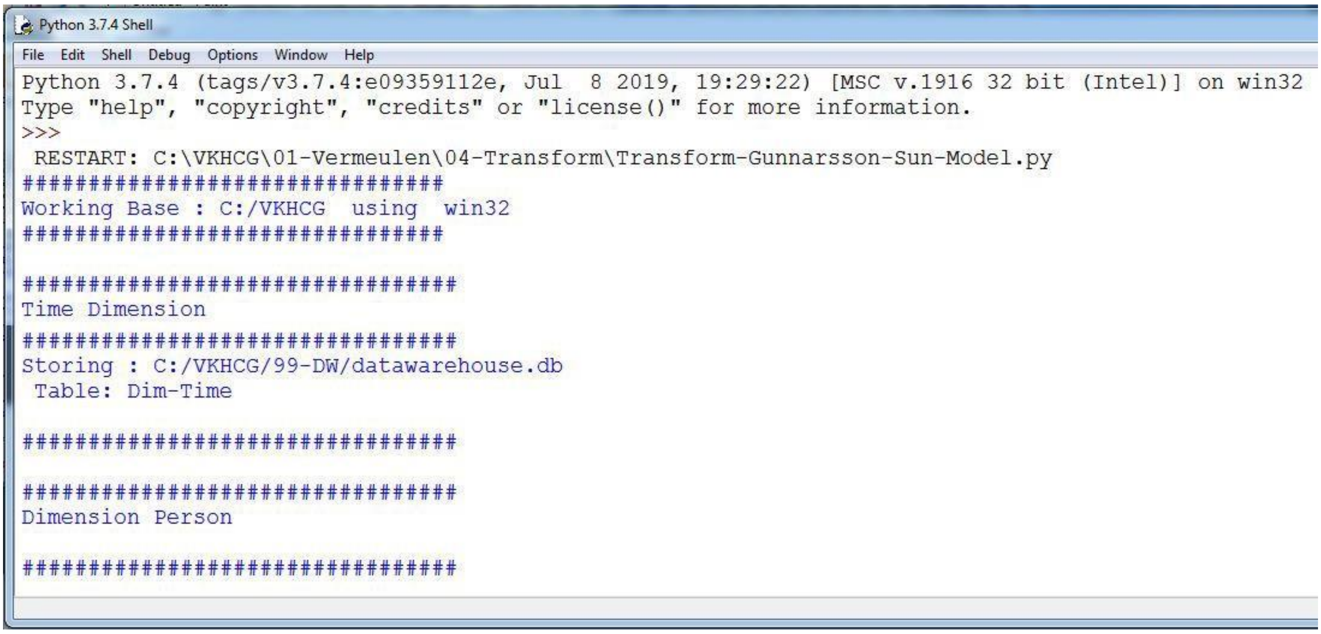
```
>>>
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson_is_Born.py
Working Base : C:/VKHCG  using  win32
Time Category
UTC Time
1960-12-20 10:15:00 (UTC) (+0000)
###################################
Birth Date in Reykjavik :
1960-12-20 09:15:00 (-01) (-0100)
###################################
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Time-Gunnarsson
###################################
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Satellite-Time-Atlantic-Reykjavik-Gunnarsson
###################################
Person Category
Name: Guðmundur Gunnarsson
Birth Date: 1960-12-20 09:15:00
Birth Zone: Atlantic/Reykjavik
UTC Birth Date: 1960-12-20 10:15:00
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Person-Gunnarsson
###################################
```

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

```
####################################################################
# -*- coding: utf-8 -*-
####################################################################
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
####################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
```

```
else:
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='01-Vermeulen'
################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir):
os.makedirs(sDataWarehousetDir)
################################################################
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
conn2 = sq.connect(sDatabaseName)
################################################################
print('\n#################################')
print('Time Dimension')
BirthZone = 'Atlantic/Reykjavik'
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [IDTimeNumber]),
('UTCDate', [BirthDateZoneStr]),
('LocalTime', [BirthDateLocal]),
('TimeZone', [BirthZone])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
################################################################
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
################################################################
sTable = 'Dim-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
print('\n#################################')
print('Dimension Person')
print('\n#################################')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
###############################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [IDPersonNumber]),
('FirstName', [FirstName]),
('LastName', [LastName]),
('Zone', ['UTC']),
('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
################################################################
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
print('\n#################################')
print('Fact - Person - time')
print('\n#################################')
IDFactNumber=str(uuid.uuid4())
PersonTimeLine=[('IDNumber', [IDFactNumber]),
('IDPersonNumber', [IDPersonNumber]),
('IDTimeNumber', [IDTimeNumber])]
PersonTimeFrame = pd.DataFrame.from_items(PersonTimeLine)
################################################################
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
################################################################
sTable = 'Fact-Person-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
```

Output:



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson-Sun-Model.py
################################
Working Base : C:/VKHCG  using  win32
################################

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

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

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

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

Building a Data Warehouse

Open the Transform-Sun-Models.py file from directory C:\VKHCG\01-Vermeulen\04-Transform.

```
######################################################################
# -*- coding: utf-8 -*-
######################################################################
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
######################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG'
print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
######################################################################
Company='01-Vermeulen'
######################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
######################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
######################################################################
```

```
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
os.makedirs(sDataVaultDir)
#################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
#################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
#################################################################
sSQL=" SELECT DateTimeValue FROM [Hub-Time];"
DateDataRaw=pd.read_sql_query(sSQL, conn2)
DateData=DateDataRaw.head(1000)
print(DateData)
#################################################################
print('\n##################################')
print('Time Dimension')
print('\n##################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
BirthZone = ('Atlantic/Reykjavik','Europe/London','UCT')
for j in range(len(BirthZone)):
t+=1
print(t,mt*3)
BirthDateUTC = datetime.strptime(DateData['DateTimeValue'][i],"%Y-%m-%d %H:%M:%S")
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone[j]))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
#################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [str(IDTimeNumber)]),
('UTCDate', [str(BirthDateZoneStr)]),
('LocalTime', [str(BirthDateLocal)]),
('TimeZone', [str(BirthZone)])]
if t==1:
TimeFrame = pd.DataFrame.from_items(TimeLine)
else:
TimeRow = pd.DataFrame.from_items(TimeLine)
TimeFrame=TimeFrame.append(TimeRow)
#################################################################
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
#################################################################
```

```
sTable = 'Dim-Time'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn3, if_exists="replace")
####################################################################
sSQL=" SELECT " + \
" FirstName," + \
" SecondName," + \
" LastName," + \
" BirthDateKey " + \
" FROM [Hub-Person];"
PersonDataRaw=pd.read_sql_query(sSQL, conn2)
PersonData=PersonDataRaw.head(1000)
####################################################################
print('\n###################################')
print('Dimension Person')
print('\n###################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
t+=1
print(t,mt)
FirstName = str(PersonData["FirstName"])
SecondName = str(PersonData["SecondName"])
if len(SecondName) > 0:
SecondName=""
LastName = str(PersonData["LastName"])
BirthDateKey = str(PersonData["BirthDateKey"])
###################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [str(IDPersonNumber)]),
('FirstName', [FirstName]),
('SecondName', [SecondName]),
('LastName', [LastName]),
('Zone', [str('UTC')]),
('BirthDate', [BirthDateKey])]
if t==1:
PersonFrame = pd.DataFrame.from_items(PersonLine)
else:
PersonRow = pd.DataFrame.from_items(PersonLine)
PersonFrame = PersonFrame.append(PersonRow)
####################################################################
DimPerson=PersonFrame
print(DimPerson)
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
####################################################################
sTable = 'Dim-Person'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
###################################################################
```

Output:
You have successfully performed data vault to data warehouse transformation.

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 variable
b = slope of the line
a = intercept (the value of y when x = 0)

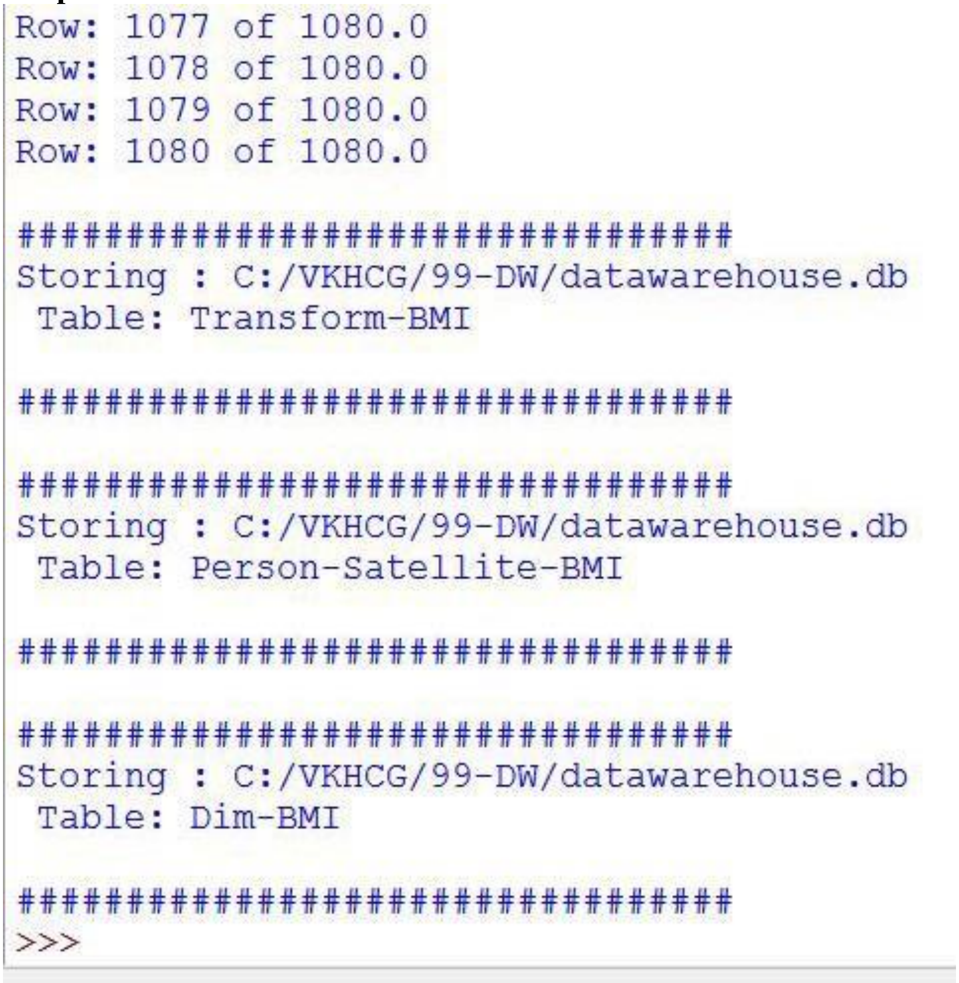
```
####################################################################
# -*- coding: utf-8 -*-
####################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
####################################################################
####################################################################
Company='01-Vermeulen'
####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
####################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
```

```
os.makedirs(sDataVaultDir)
################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
################################################################
t=0
tMax=((300-100)/10)*((300-30)/5)
for heightSelect in range(100,300,10):
for weightSelect in range(30,300,5):
height = round(heightSelect/100,3)
weight = int(weightSelect)
bmi = weight/(height*height)
if bmi <= 18.5:
BMI_Result=1
elif bmi > 18.5 and bmi < 25:
BMI_Result=2
elif bmi > 25 and bmi < 30:
BMI_Result=3
elif bmi > 30:
BMI_Result=4
else:
BMI_Result=0
PersonLine=[('PersonID', [str(t)]),
('Height', [height]),
('Weight', [weight]),
('bmi', [bmi]),
('Indicator', [BMI_Result])]
t+=1
print('Row:',t,'of',tMax)
if t==1:
PersonFrame = pd.DataFrame.from_items(PersonLine)
else:
PersonRow = pd.DataFrame.from_items(PersonLine)
PersonFrame = PersonFrame.append(PersonRow)
################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
################################################################
sTable = 'Transform-BMI'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
################################################################
```

```
############################################################################
sTable = 'Person-Satellite-BMI'
print('\n####################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n####################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
############################################################################
############################################################################
sTable = 'Dim-BMI'
print('\n####################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n####################################')
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
############################################################################
fig = plt.figure()
PlotPerson=DimPerson[DimPerson['Indicator']==1]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, ".")
PlotPerson=DimPerson[DimPerson['Indicator']==2]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "o")
PlotPerson=DimPerson[DimPerson['Indicator']==3]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "+")
PlotPerson=DimPerson[DimPerson['Indicator']==4]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "^")
plt.axis('tight')
plt.title("BMI Curve")
plt.xlabel("Height(meters)")
plt.ylabel("Weight(kg)")
plt.plot()
# Load the diabetes dataset
diabetes = datasets.load_diabetes()
# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-50:]
diabetes_y_train = diabetes.target[:-30]
diabetes_y_test = diabetes.target[-50:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print('Coefficients: \n', regr.coef_)
print("Mean squared error: %.2f"
% mean_squared_error(diabetes_y_test, diabetes_y_pred))
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))
```

```
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.axis('tight')
plt.title("Diabetes")
plt.xlabel("BMI")
plt.ylabel("Age")
plt.show()
```

Output:



```
Row: 1077 of 1080.0
Row: 1078 of 1080.0
Row: 1079 of 1080.0
Row: 1080 of 1080.0

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

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

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

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

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

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

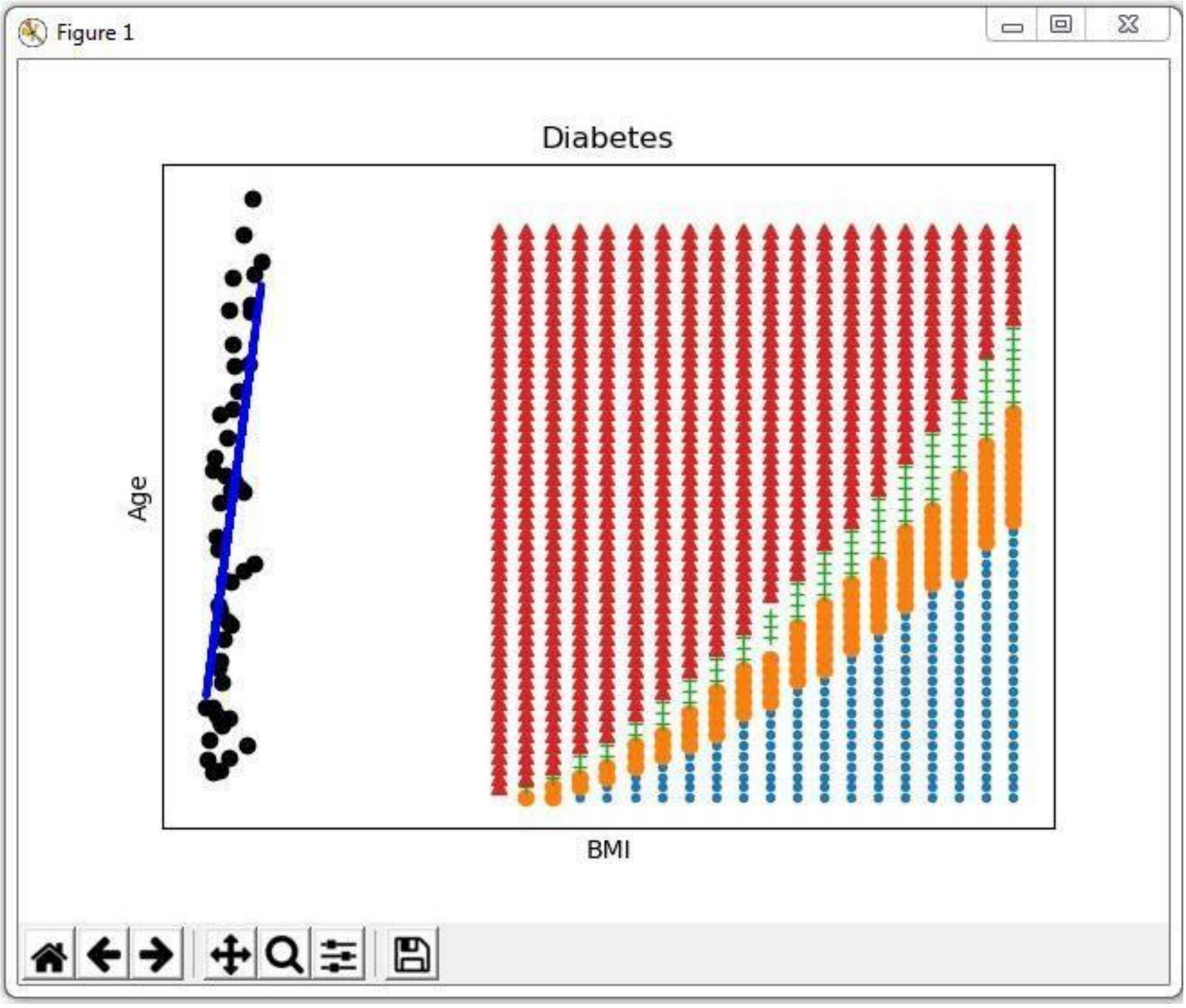


Figure 1
Diabetes
Age
BMI

Practical 7
Organizing Data

Organize Superstep

The Organize superstep takes the complete data warehouse you built at the end of the Transform superstep and
subsections it into business-specific data marts. A data mart is the access layer of the data warehouse environment built to expose data to the users. The data mart is a subset of the data warehouse and is generally
oriented to a specific business group.

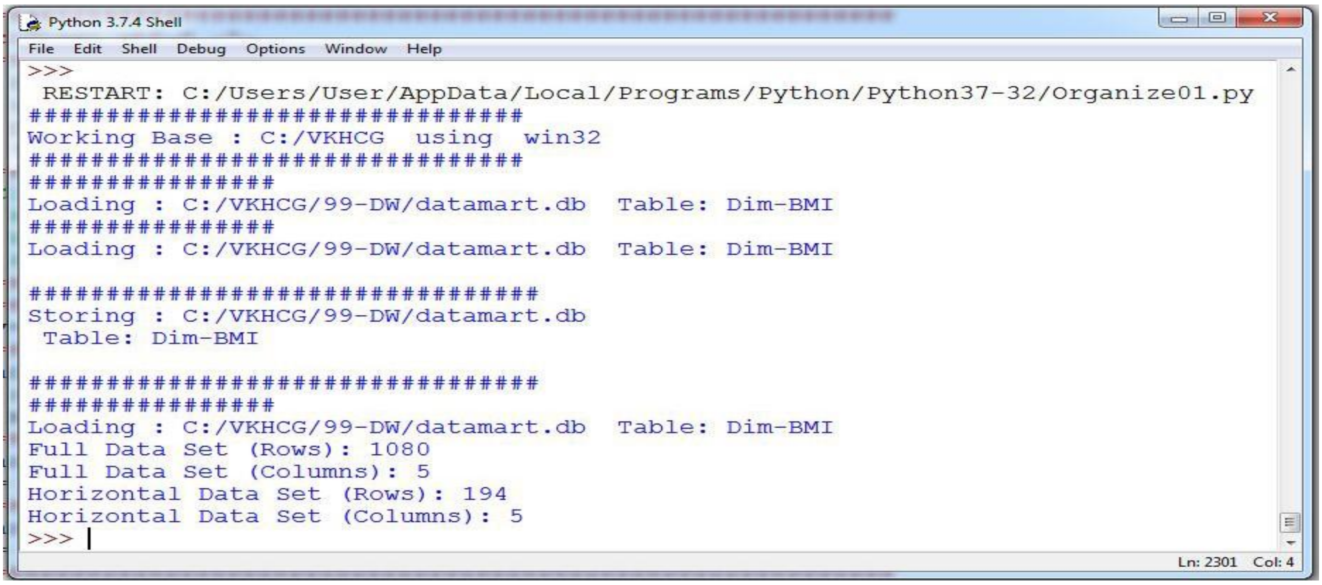
Horizontal Style

Performing horizontal-style slicing or subsetting of the data warehouse is achieved by applying a filter technique that forces the data warehouse to show only the data for a specific preselected set of filtered outcomes against the data population. The horizontal-style slicing selects the subset of rows from the population while preserving the columns. That is, the data science tool can see the complete record for the records in the subset of records.

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

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

Output:



```
>>>
 RESTART: C:/Users/User/AppData/Local/Programs/Python/Python37-32/Organize01.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

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

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

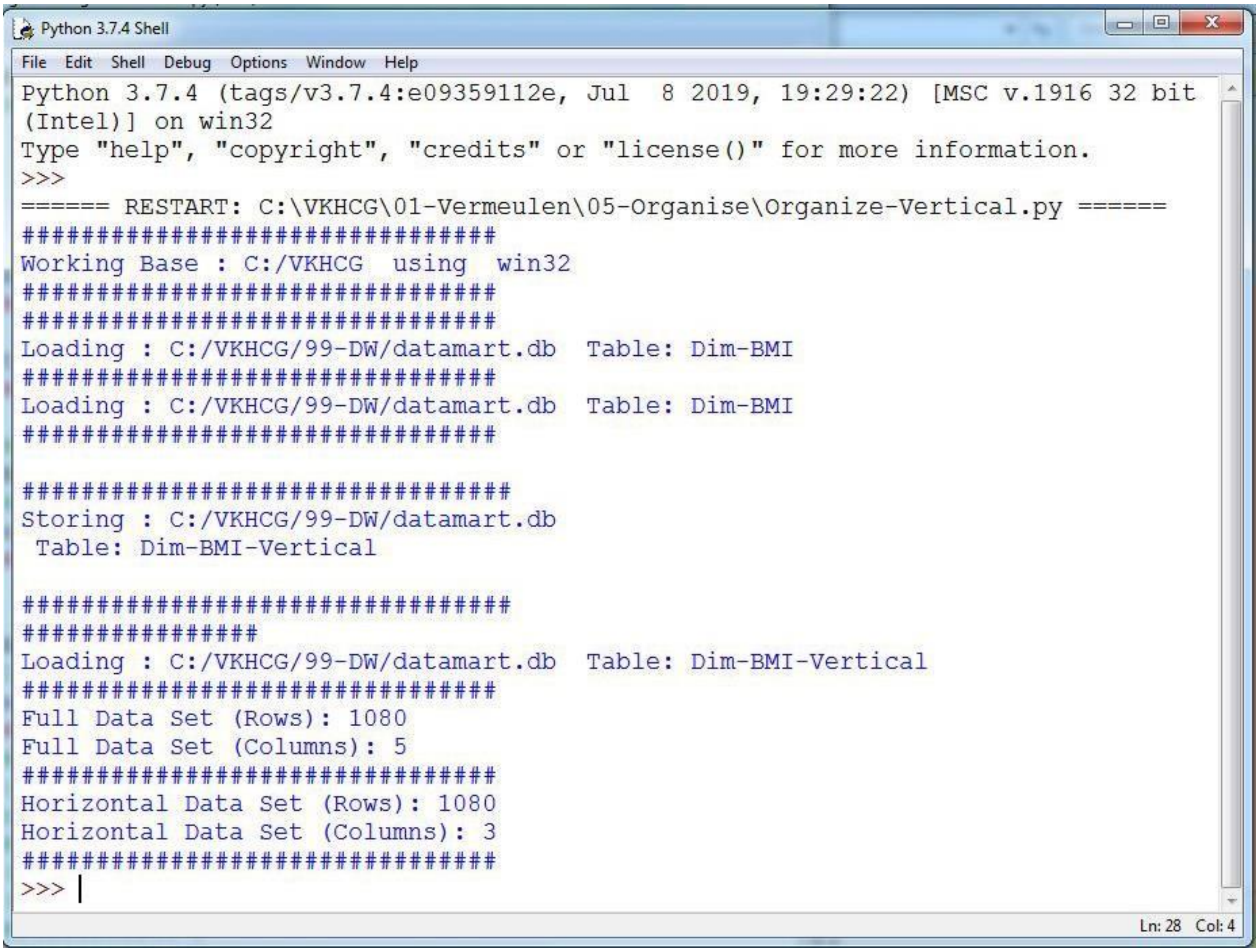
Vertical Style

Performing vertical-style slicing or subsetting of the data warehouse is achieved by applying a filter technique
that forces the data warehouse to show only the data for specific preselected filtered outcomes against the data
population. The vertical-style slicing selects the subset of columns from the population, while preserving the
rows. That is, the data science tool can see only the preselected columns from a record for all the records in the
population.

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

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



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Vertical.py ======
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################

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

#################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Vertical
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 1080
Horizontal Data Set (Columns): 3
################################
>>> 
```

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

Island Style

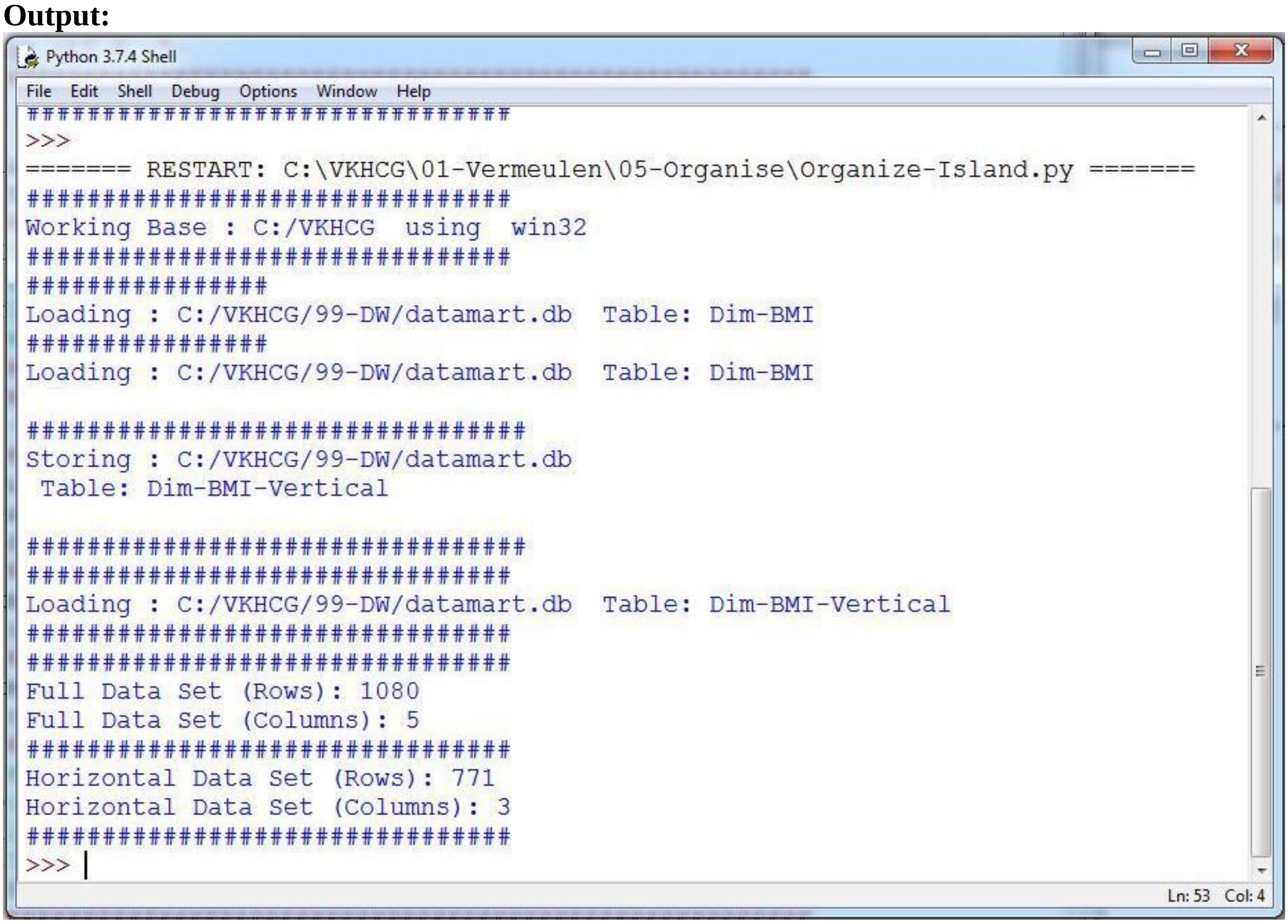
Performing island-style slicing or subsetting of the data warehouse is achieved by applying a combination of
horizontal- and vertical-style slicing. This generates a subset of specific rows and specific columns reduced at
the same time.

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

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

```
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
Height,\
Weight,\
Indicator\
FROM [Dim-BMI]\
WHERE Indicator > 2\
ORDER BY \
Height,\
Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
################################################################
sTable = 'Dim-BMI-Vertical'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
print('################################')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('##################################')
################################################################################
```

Output:



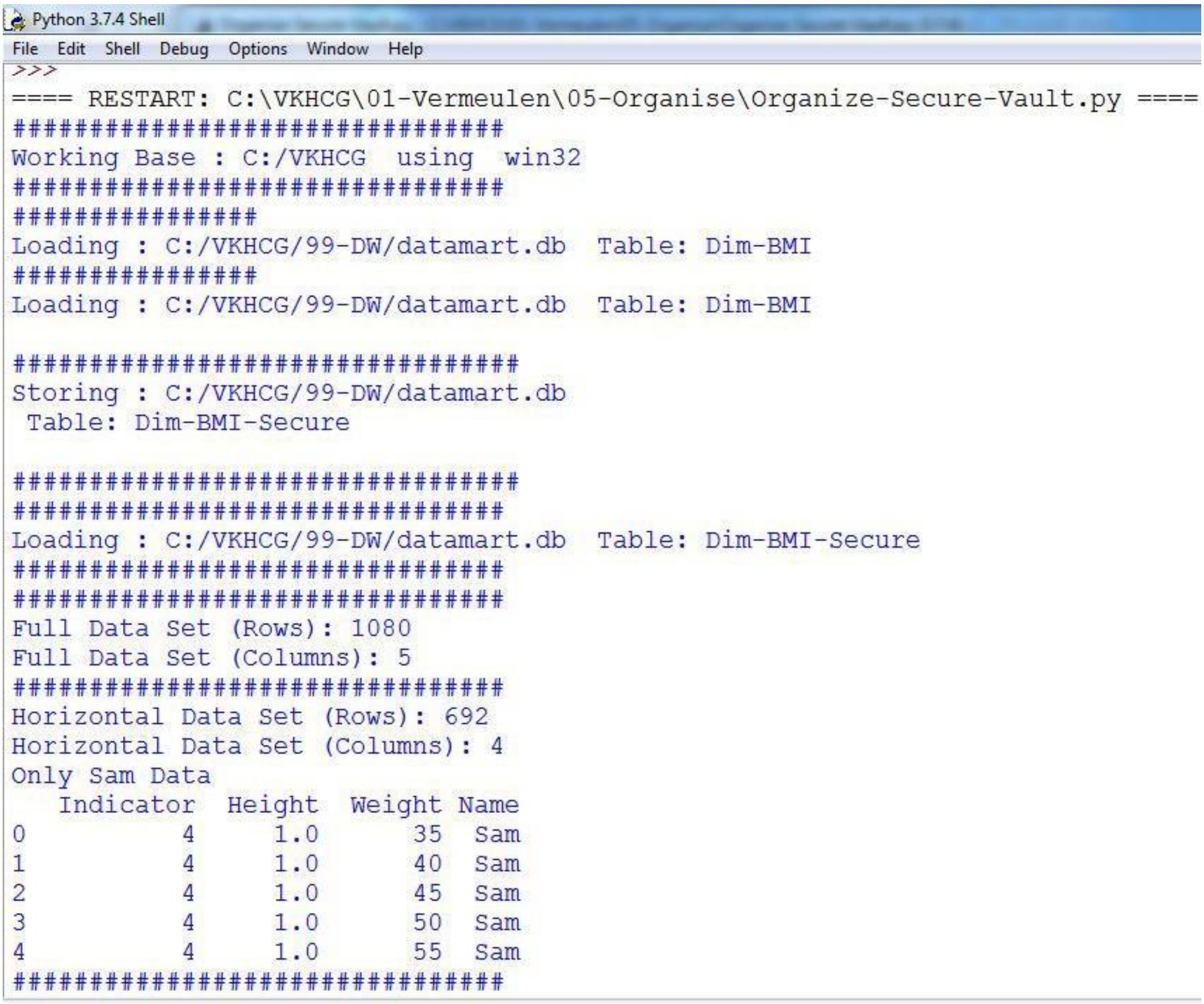
```
##################################
>>>
======= RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Island.py =======
##################################
Working Base : C:/VKHCG  using  win32
##################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

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

###################################
##################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Vertical
##################################
##################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
##################################
Horizontal Data Set (Rows): 771
Horizontal Data Set (Columns): 3
##################################
>>> |
```

Secure Vault Style
The secure vault is a version of one of the horizontal, vertical, or island slicing techniques, but the outcome is
also attached to the person who performs the query. This is common in multi-security environments, where
different users are allowed to see different data sets.
This process works well, if you use a role-based access control (RBAC) approach to restricting system access
to authorized users. The security is applied against the “role,” and a person can then, by the security system,
simply be added or removed from the role, to enable or disable access.
C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Secure-Vault.py

```
################################################################################
# -*- coding: utf-8 -*-
################################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
################################################################
Company='01-Vermeulen'
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
Height,\
Weight,\
Indicator,\
CASE Indicator\
WHEN 1 THEN 'Pip'\
WHEN 2 THEN 'Norman'\
WHEN 3 THEN 'Grant'\
ELSE 'Sam'\
END AS Name\
FROM [Dim-BMI]\
WHERE Indicator > 2\
ORDER BY \
Height,\
Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
################################################################
sTable = 'Dim-BMI-Secure'
```

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

Output:



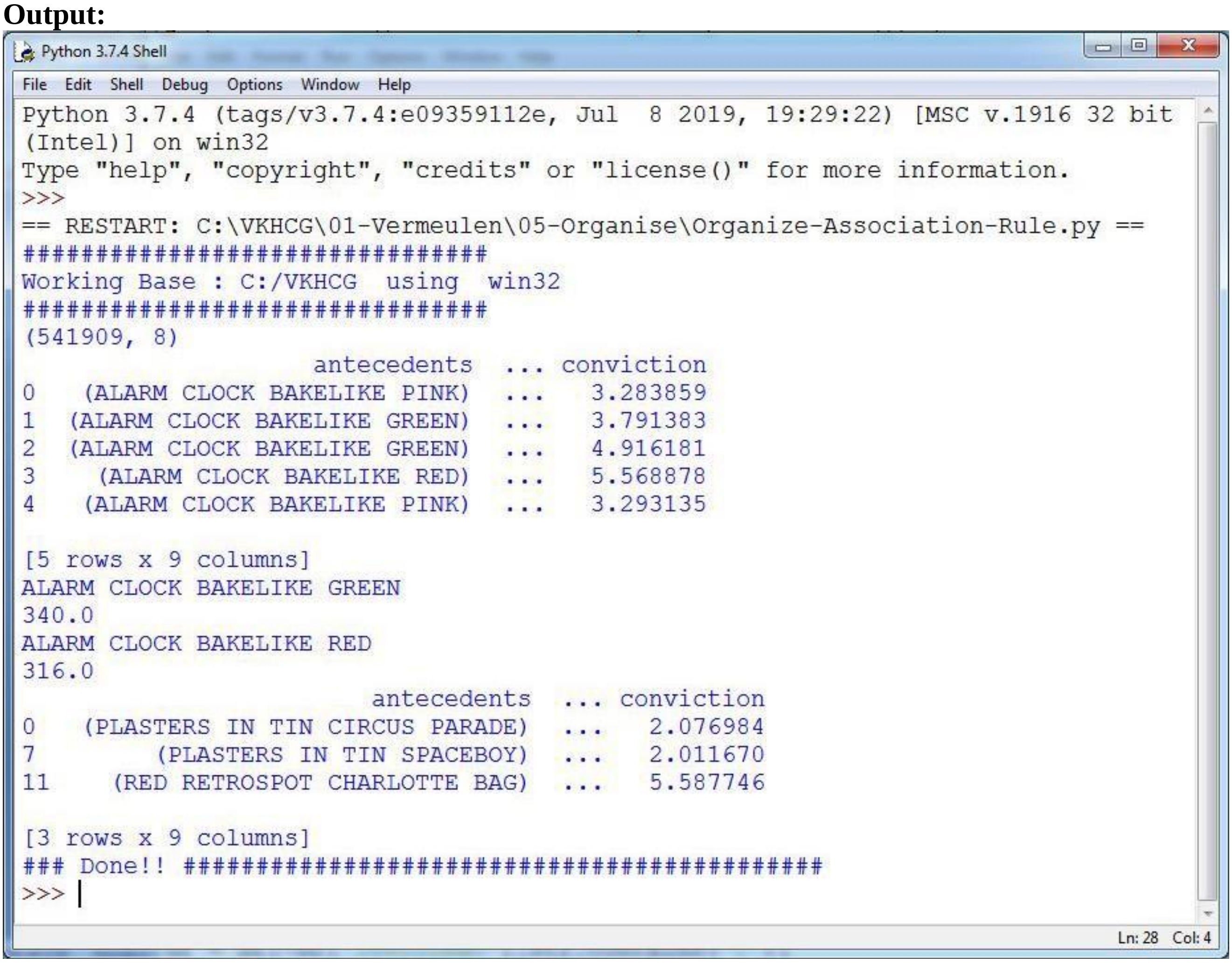
```
>>>
==== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Secure-Vault.py ====
################################
Working Base : C:/VKHCG  using  win32
################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

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

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

Association Rule Mining
Association rule learning is a rule-based machine-learning method for discoveringinteresting relations between
variables in large databases, similar to the data you willfind in a data lake. The technique enables you to investigate the interaction between datawithin the same population. Lift is simply estimatedby the ratio of the
joint probability of two items x and y, divided by the product of theirindividual probabilities:
C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Association-Rule.py

```
########################################################################
# -*- coding: utf-8 -*-
########################################################################
import sys
import os
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
########################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='01-Vermeulen'
InputFileName='Online-Retail-Billboard.xlsx'
EDSAssessDir='02-Assess/01-EDS'
InputAssessDir=EDSAssessDir + '/02-Python'
################################################################
sFileAssessDir=Base + '/' + Company + '/' + InputAssessDir
if not os.path.exists(sFileAssessDir):
os.makedirs(sFileAssessDir)
################################################################
sFileName=Base+'/'+ Company + '/00-RawData/' + InputFileName
################################################################
df = pd.read_excel(sFileName)
print(df.shape)
################################################################
df['Description'] = df['Description'].str.strip()
df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]
basket = (df[df['Country'] =="France"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
################################################################
def encode_units(x):
if x <= 0:
return 0
if x >= 1:
return 1
################################################################
basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
print(rules.head())
rules[ (rules['lift'] >= 6) &
(rules['confidence'] >= 0.8) ]
################################################################
sProduct1='ALARM CLOCK BAKELIKE GREEN'
print(sProduct1)
print(basket[sProduct1].sum())
sProduct2='ALARM CLOCK BAKELIKE RED'
print(sProduct2)
print(basket[sProduct2].sum())
################################################################
basket2 = (df[df['Country'] =="Germany"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
basket_sets2 = basket2.applymap(encode_units)
```

```
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)
print(rules2[ (rules2['lift'] >= 4) &
(rules2['confidence'] >= 0.5)])
################################################################
print('### Done!! ############################################')
################################################################
```

Output:



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Association-Rule.py ==
################################
Working Base : C:/VKHCG  using  win32
################################
(541909, 8)
                     antecedents  ... conviction
0   (ALARM CLOCK BAKELIKE PINK)  ...   3.283859
1  (ALARM CLOCK BAKELIKE GREEN)  ...   3.791383
2  (ALARM CLOCK BAKELIKE GREEN)  ...   4.916181
3    (ALARM CLOCK BAKELIKE RED)  ...   5.568878
4   (ALARM CLOCK BAKELIKE PINK)  ...   3.293135

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

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

Create a Network Routing Diagram

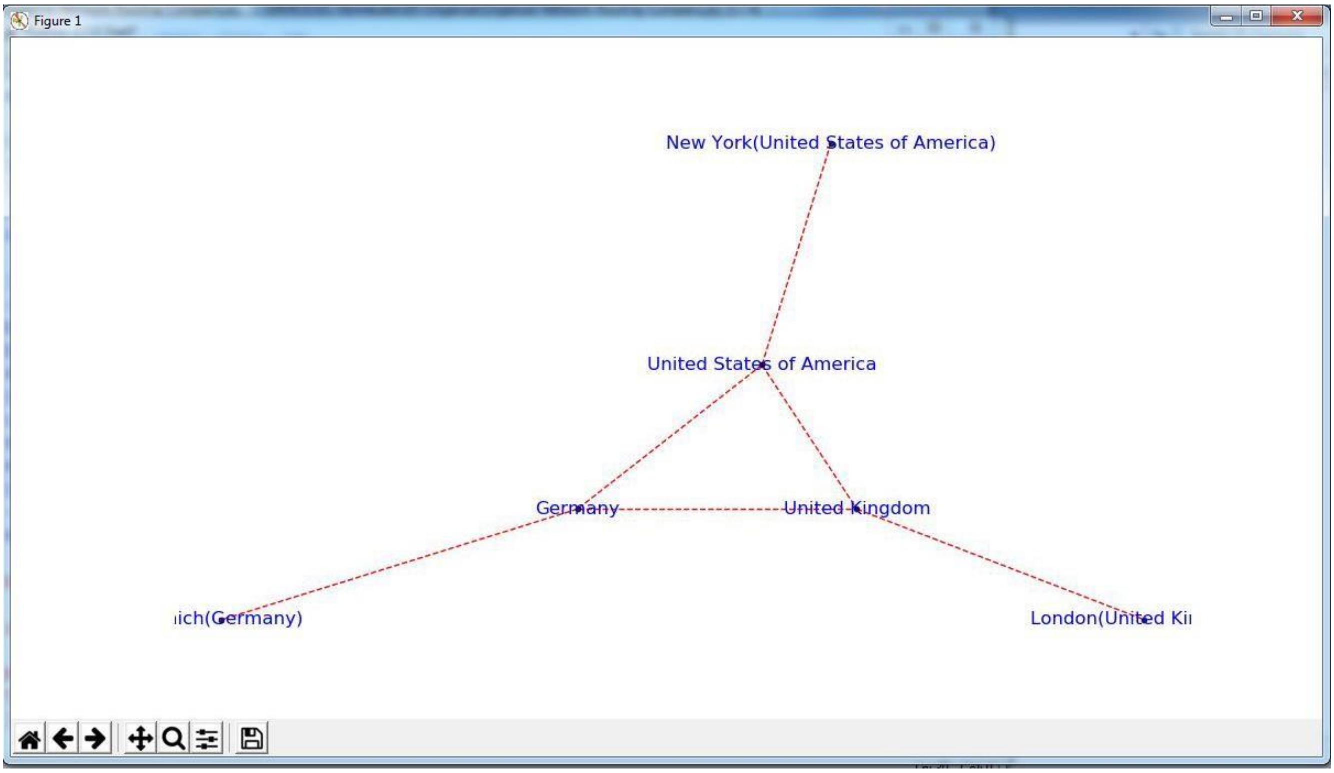
I will guide you through a possible solution for the requirement, by constructing an island-style Organize superstep that uses a graph data model to reduce the records and the columns on the data set.

C:\VKHCG\01-Vermeulen\05-Organise\ Organise-Network-Routing-Company.py

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

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

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



Picking Content for Billboards

To enable the marketing salespeople to sell billboard content, they will require a diagram to show what billboards connect to which office content publisher. Each of Krennwallner's billboards has a proximity sensor that enables the content managers to record when a registered visitor points his/her smartphone at the billboard
content or touches the near-field pad with a mobile phone.

Program will assist you in building an organized graph of the billboards' locations data to help you to gain insights into the billboard locations and content picking process.

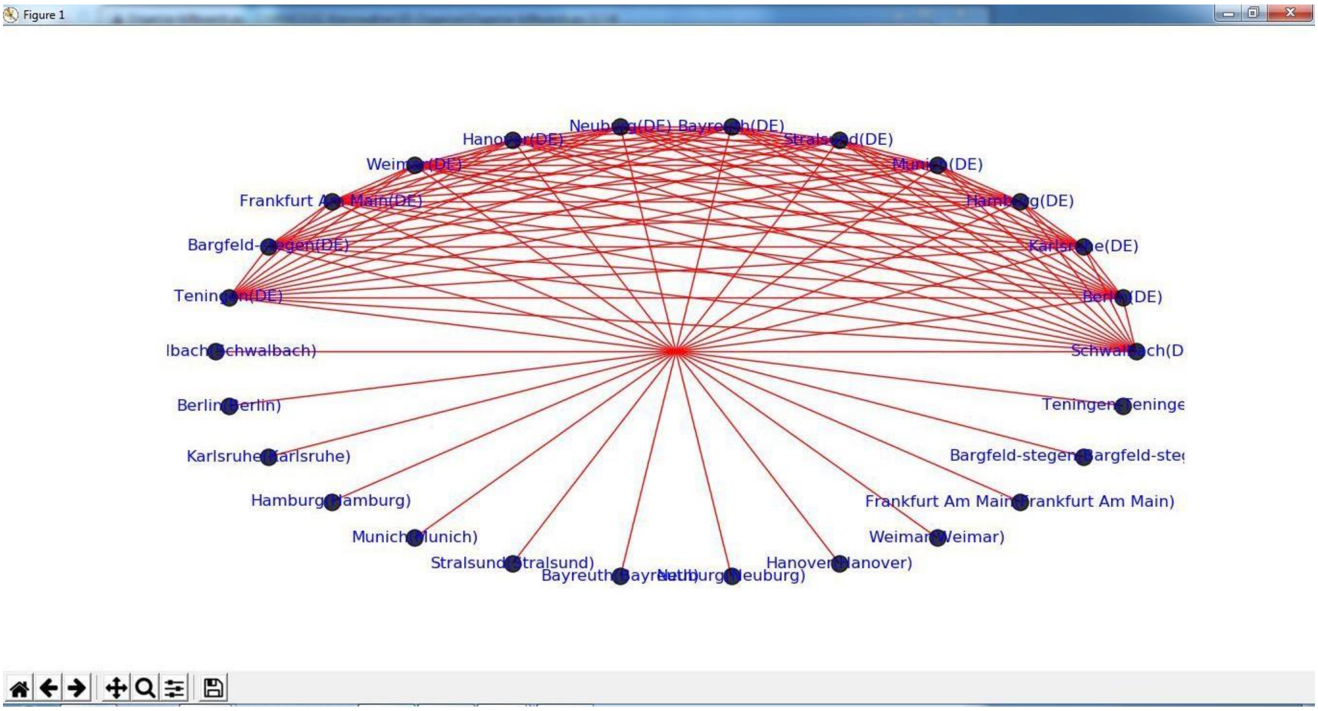
C:\VKHCG\02-Krennwallner\05-Organise\ Organise-billboards.py

```
########################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
```

```
########################################################################
pd.options.mode.chained_assignment = None
########################################################################
Base='C:/VKHCG'
########################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
########################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-DE-Billboard-Visitor.csv'
########################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Billboards.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Billboards.png'
Company='02-Krennwallner'
########################################################################
########################################################################
### Import Company Data
########################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
BillboardDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('#################################')
########################################################################
print(BillboardDataRaw.head())
print(BillboardDataRaw.shape)
BillboardData=BillboardDataRaw
sSample=list(np.random.choice(BillboardData.shape[0],20))
#######################################################################
G=nx.Graph()
for i in sSample:
for j in sSample:
Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['BillboardCountry'][i] + ')'
Node1=BillboardData['BillboardPlaceName'][j] + '('+ BillboardData['BillboardCountry'][i] + ')'
if Node0 != Node1:
G.add_edge(Node0,Node1)
for i in sSample:
Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorPlaceName'][i] + ')'
Node1=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorCountry'][i] + ')'
if Node0 != Node1:
G.add_edge(Node0,Node1)
print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
########################################################################
sFileName=Base + '/02-Krennwallner/' + sOutputFileName1
print('#################################')
print('Storing :',sFileName)
print('#################################')
nx.write_gml(G, sFileName)
########################################################################
```

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

Output :



Create a Delivery Route

Hillman requires a new delivery route plan from HQ-KA13’s delivery region. Themanaging director has to
know the following:

• What his most expensive route is, if the cost is £1.50 per mile and twotrips are planned per day

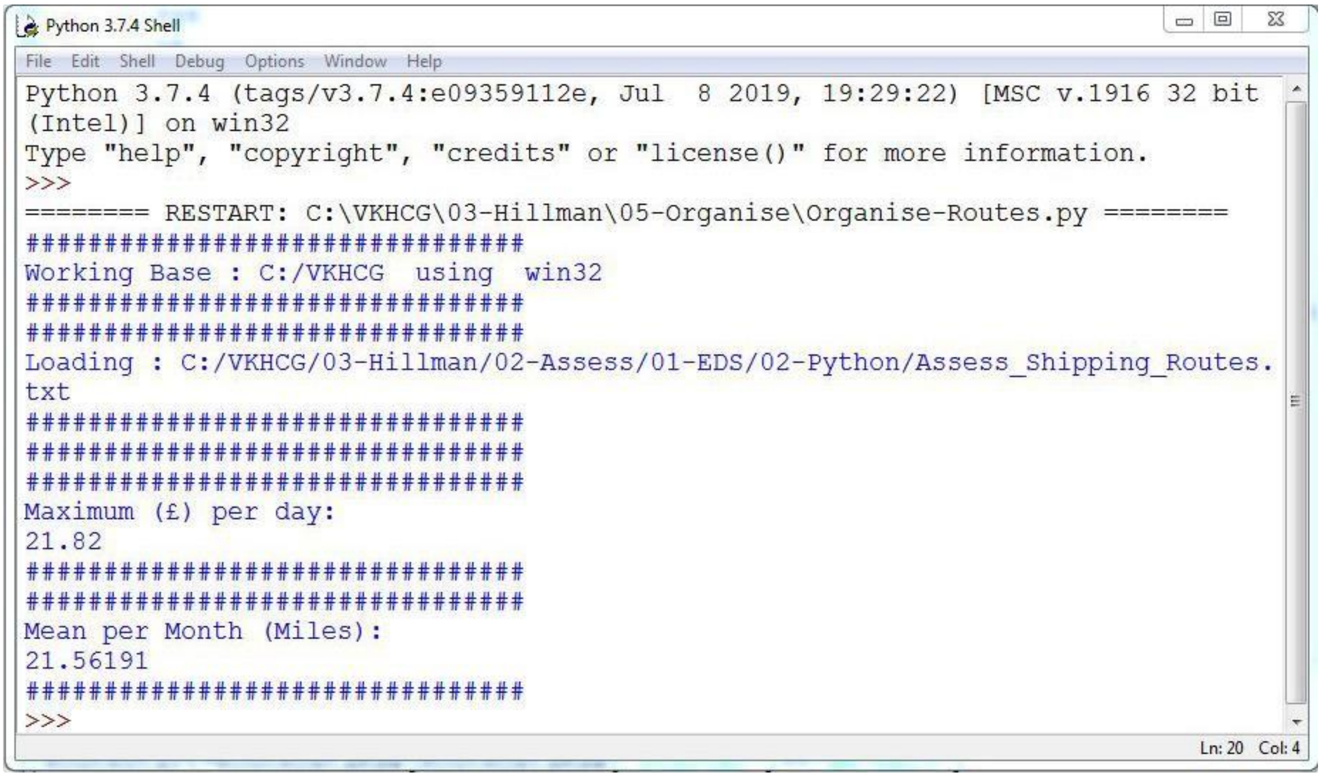
• What the average travel distance in miles is for the region per 30-daymonth

With your newfound knowledge in building the technology stack for turning datalakes into business assets, can
you convert the graph stored in the Assess step called
“Assess_Best_Logistics” into the shortest path between the two points?

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

Output:



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
======== RESTART: C:\VKHCG\03-Hillman\05-Organise\Organise-Routes.py ========
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/03-Hillman/02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.
txt
################################
################################
################################
Maximum (£) per day:
21.82
################################
################################
Mean per Month (Miles):
21.56191
################################
>>> 
```

Clark Ltd

Our financial services company has been tasked to investigate the options to convert1 million pounds sterling
into extra income. Mr. Clark Junior suggests using the simplevariance in the daily rate between the British pound sterling and the US dollar, togenerate extra income from trading. Your chief financial officer wants to
know if this isfeasible?

Simple Forex Trading Planner

Your challenge is to take 1 million US dollars or just over six hunderd thou sand pounds sterling and, by simply converting it between pounds sterling and US dollars, achieve a profit. Are you up to this challenge?

The Program will help you how to model this problem and achieve a positive outcome. The forex data has been collected on a daily basis by Clark's accounting department, from previous overseas transactions.

C:\VKHCG\04-Clark\05-Organise\Organise-Forex.py

```
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
import re
################################################################
Base='C:/VKHCG'
################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
```

```
sInputFileName='03-Process/01-EDS/02-Python/Process_ExchangeRates.csv'
####################################################################
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Forex.csv'
Company='04-Clark'
####################################################################
sDatabaseName=Base + '/' + Company + '/05-Organise/SQLite/clark.db'
conn = sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
####################################################################
####################################################################
### Import Forex Data
####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('#################################')
####################################################################
ForexDataRaw.index.names = ['RowID']
sTable='Forex_All'
print('Storing :',sDatabaseName,' Table:',sTable)
ForexDataRaw.to_sql(sTable, conn, if_exists="replace")
####################################################################
sSQL="SELECT 1 as Bag\
, CAST(min(Date) AS VARCHAR(10)) as Date \
,CAST(1000000.0000000 as NUMERIC(12,4)) as Money \
,'USD' as Currency \
FROM Forex_All \
;"
sSQL=re.sub("\s\s+", " ", sSQL)
nMoney=pd.read_sql_query(sSQL, conn)
####################################################################
nMoney.index.names = ['RowID']
sTable='MoneyData'
print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
####################################################################
sTable='TransactionData'
print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
####################################################################
ForexDay=pd.read_sql_query("SELECT Date FROM Forex_All GROUP BY Date;", conn)
####################################################################
t=0
for i in range(ForexDay.shape[0]):
sDay1=ForexDay['Date'][i]
sDay=str(sDay1)
sSQL='\
SELECT M.Bag as Bag, \
F.Date as Date, \
round(M.Money * F.Rate,6) AS Money, \
F.CodeIn AS PCurrency, \
F.CodeOut AS Currency \
FROM MoneyData AS M \
JOIN \
( \
SELECT \
CodeIn, CodeOut, Date, Rate \
FROM \
Forex_All \
WHERE\
CodeIn = "USD" AND CodeOut = "GBP" \
UNION \
SELECT \
CodeOut AS CodeIn, CodeIn AS CodeOut, Date, (1/Rate) AS Rate \
FROM \
Forex_All \
WHERE\
CodeIn = "USD" AND CodeOut = "GBP" \
) AS F \
ON \
M.Currency=F.CodeIn \
AND \
F.Date ="' +sDay + '";'
sSQL=re.sub("\s\s+", " ", sSQL)
ForexDayRate=pd.read_sql_query(sSQL, conn)
for j in range(ForexDayRate.shape[0]):
sBag=str(ForexDayRate['Bag'][j])
nMoney=str(round(ForexDayRate['Money'][j],2))
sCodeIn=ForexDayRate['PCurrency'][j]
sCodeOut=ForexDayRate['Currency'][j]
sSQL='UPDATE MoneyData SET Date= "' + sDay + '", '
sSQL= sSQL + ' Money = ' + nMoney + ', Currency="' + sCodeOut + '"'
sSQL= sSQL + ' WHERE Bag=' + sBag + ' AND Currency="' + sCodeIn + '";'
sSQL=re.sub("\s\s+", " ", sSQL)
cur = conn.cursor()
cur.execute(sSQL)
conn.commit()
t+=1
print('Trade :', t, sDay, sCodeOut, nMoney)
sSQL=' \
INSERT INTO TransactionData ( \
RowID, \
Bag, \
Date, \
Money, \
Currency \
) \
SELECT ' + str(t) + ' AS RowID, \
Bag, \
Date, \
Money, \
Currency \
FROM MoneyData \
;'
sSQL=re.sub("\s\s+", " ", sSQL)
cur = conn.cursor()
cur.execute(sSQL)
conn.commit()
########################################################################
sSQL="SELECT RowID, Bag, Date, Money, Currency FROM TransactionData ORDER BY
RowID;"
sSQL=re.sub("\s\s+", " ", sSQL)
TransactionData=pd.read_sql_query(sSQL, conn)
OutputFile=Base + '/' + Company + '/' + sOutputFileName
TransactionData.to_csv(OutputFile, index = False)
########################################################################
```

Output:

Save the Assess-Forex.py file, then compile and execute with your Python compiler.
This will produce a set of demonstrated values onscreen.