DATA SCIENCE

&

SOFT COMPUTING TECHNIQUES

Submitted for the award the Degree of

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY) PART 1

By

RAJ NITIN DHANMEHER

DEPARTMENT OF INFORMATION TECHNOLOGY

VIDYA VARDHINIS ANNASAHEB VARTAK COLLEGE OF ARTS. K.M.

COLLEGE OF COMMERCE, E.S.A COLLEGE OF SCIENCE

AFFILIATED TO UNIVERSITY OF MUMBAI)

VASAI (WEST)-401202,

DIST PALGHAR MAHARASHTRA 2025-2026

DATA SCIENCE

DATA SCIENCE

Submitted for the award the Degree of

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY) PART 1

By

RAJ NITIN DHANMEHER

DEPARTMENT OF INFORMATION TECHNOLOGY

VIDYA VARDHINIS ANNASAHEB VARTAK COLLEGE OF ARTS. K.M. COLLEGE OF COMMERCE, E.S.A COLLEGE OF SCIENCE

AFFILIATED TO UNIVERSITY OF MUMBAI)

VASAI (WEST)-401 202,

DIST PALGHAR MAHARASHTRA 2025-2026

DEPARTMENT OF INFORMATION TECHNOLOGY

VIDYAVARDHINIS ANNASAHEB VARTAK COLLEGE OF ARTS. K.M. OF COLLEGE COMMERCE, E.S.A COLLEGE OF SCIENCE AFFILIATED OUNIVERSITYOF MUMBAI)

VASAI (WEST)-401202,

DIST PALGHAR MAHARASHTRA 2024-2025

CERTIFICATE

This is to certify that the journal entitled for **" DATA SCIENCE"** is Benefited work of

RAJ NITIN DHANMEHER Bearing Seat. No: _____Submitted for the award of degree of MASTERS OF SCIENCE in (INFORMATION TECHNOLOGY) PART 1 from University of Mumbai.

Internal Guide **Head of Department (HOD)**

External Examiner

Date: **College Seal**

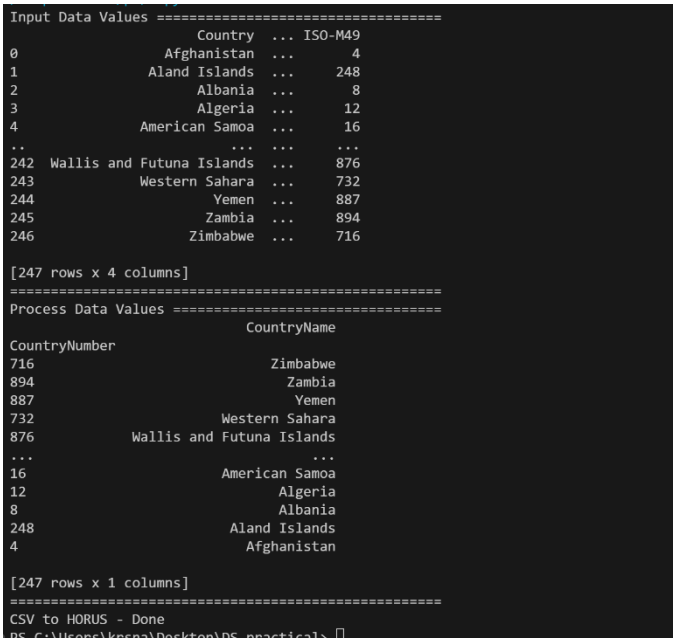
Index

Practical 1

Practical 1
Homogeneous Ontology for Recursive Uniform Schema (HORUS)

a) Text delimited CSV to HORUS format.

```
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')
```



```
Input Data Values ===================================
                        Country  ... ISO-M49
0                   Afghanistan  ...       4
1                 Aland Islands  ...     248
2                       Albania  ...       8
3                       Algeria  ...      12
4                American Samoa  ...      16
..                          ...  ...     ...
242  Wallis and Futuna Islands  ...     876
243              Western Sahara  ...     732
244                       Yemen  ...     887
245                      Zambia  ...     894
246                    Zimbabwe  ...     716

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

[247 rows x 1 columns]
=====================================================
CSV to HORUS - Done
PS C:\Users\kpsna\Desktop\DS_practical>
```

b) XML to HORUS Format

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

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

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

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

c) JSON to HORUS Format

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

```
/DS practical/p2/c.py
Input Data Values ==================================
                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0                 Afghanistan         AF        AFG        4
1               Aland Islands         AX        ALA      248
2                     Albania         AL        ALB        8
3                     Algeria         DZ        DZA       12
4              American Samoa         AS        ASM       16
..                        ...        ...        ...      ...
242  Wallis and Futuna Islands         WF        WLF      876
243             Western Sahara         EH        ESH      732
244                      Yemen         YE        YEM      887
245                     Zambia         ZM        ZMB      894
246                   Zimbabwe         ZW        ZWE      716

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

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

d) MySql Database to HORUS Format

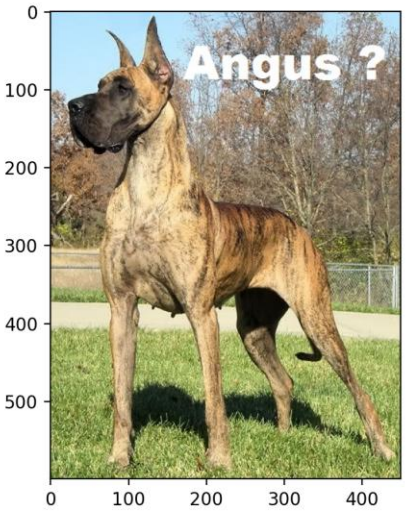
```
import pandas as pd
import sqlite3 as sq
# Input Agreement ============================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)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

```
/DS practical/p2/d.py"
Input Data Values ==================================
     index                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0        0                Afghanistan         AF        AFG        4
1        1              Aland Islands         AX        ALA      248
2        2                    Albania         AL        ALB        8
3        3                    Algeria         DZ        DZA       12
4        4             American Samoa         AS        ASM       16
..     ...                        ...        ...        ...      ...
242    242  Wallis and Futuna Islands         WF        WLF      876
243    243             Western Sahara         EH        ESH      732
244    244                      Yemen         YE        YEM      887
245    245                     Zambia         ZM        ZMB      894
246    246                   Zimbabwe         ZW        ZWE      716

[247 rows x 5 columns]
==================================================
Process Data Values ================================
              index                CountryName
CountryNumber
716             246                   Zimbabwe
894             245                     Zambia
887             244                      Yemen
732             243             Western Sahara
876             242  Wallis and Futuna Islands
...             ...                        ...
16                4             American Samoa
12                3                    Algeria
8                 2                    Albania
248               1              Aland Islands
4                 0                Afghanistan

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

e) Picture (JPEG) to HORUS Format

```
# E. Picture (JPEG) to HORUS Format
# Install the imageio library if you haven't already
#pip install imageio

import imageio
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Input Agreement ============================================
sInputFileName = 'C:/VKHCG/05-DS/9999-Data/Angus.jpg'
InputData = imageio.imread(sInputFileName)
print('Input Data Values ===================================')
print('X: ', InputData.shape[0])
print('Y: ', InputData.shape[1])
print('RGBA: ', InputData.shape[2])
print('==================================================')
# Processing Rules ===========================================
ProcessRawData = InputData.flatten()
y = InputData.shape[2] + 2
x = int(ProcessRawData.shape[0] / y)
ProcessData = pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
# Check the shape of the ProcessData dataframe
print('ProcessData shape:', ProcessData.shape)
sColumns = ['XAxis', 'YAxis', 'Red', 'Green', 'Blue', 'Alpha']
if ProcessData.shape[1] == len(sColumns):
    ProcessData.columns = sColumns
```

```
else:
   print("Column mismatch: ProcessData has", ProcessData.shape[1], "columns, expected", len(sColumns))
ProcessData.index.names = ['ID']
print('Rows: ', ProcessData.shape[0])
print('Columns :', ProcessData.shape[1])
print('====================================================')
print('Process Data Values ================================')
print('====================================================')
plt.imshow(InputData)
plt.show()
print('====================================================')

# Output Agreement ===========================================
OutputData = ProcessData
print('Storing File')
sOutputFileName = 'C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index=False)
print('====================================================')
print('Picture to HORUS - Done')
print('====================================================')
```



f) Video to HORUS Format

```
#pip install opencv-python

import os
import shutil
import cv2
# Input Agreement ============================================
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)
```

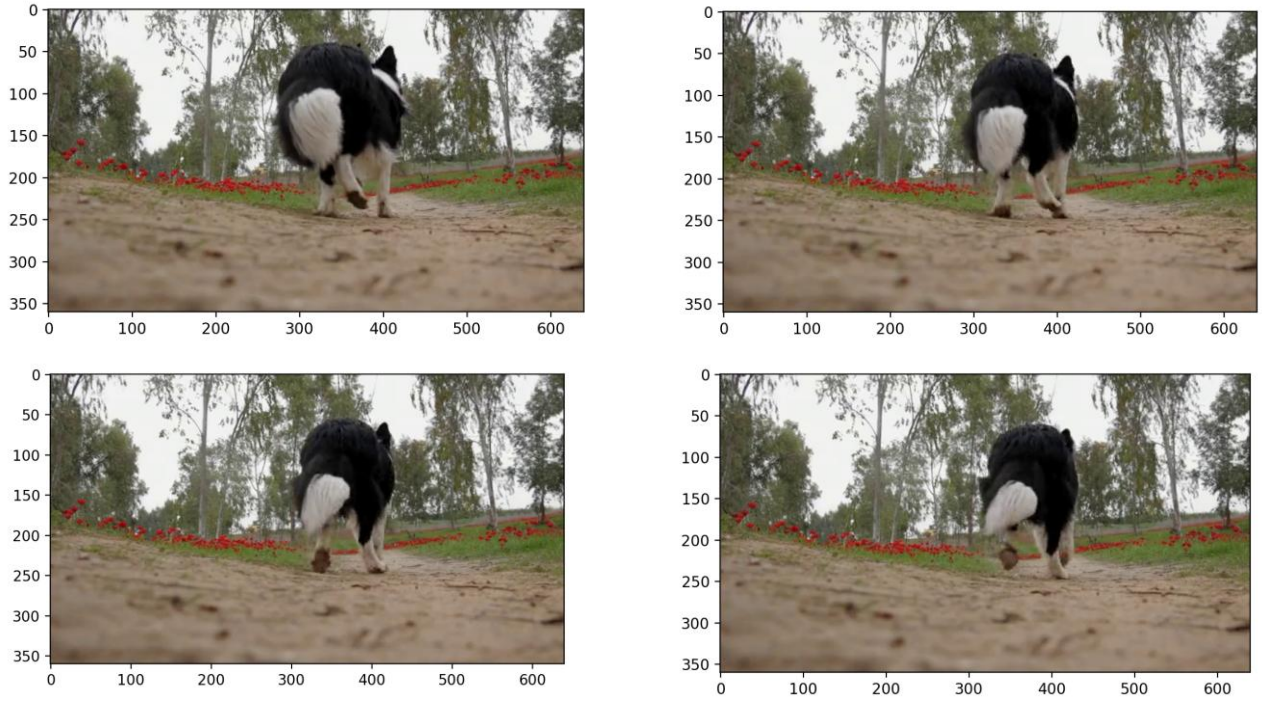
```
print('====================================================')
print('Start Movie to Frames')
print('====================================================')
vidcap = cv2.VideoCapture(sInputFileName)
success, image = vidcap.read()
count = 0
while success:
   sFrame = os.path.join(sDataBaseDir, 'dog-frame-{:04d}.jpg'.format(count))
   print('Extracted: ', sFrame)
   cv2.imwrite(sFrame, image)
   success, image = vidcap.read()
   if os.path.exists(sFrame) and os.path.getsize(sFrame) == 0:
      os.remove(sFrame)
      print('Removed: ', sFrame)
      count -= 1
   if cv2.waitKey(10) == 27:  # exit if Escape is hit
      break
   count += 1
print('====================================================')
print('Generated : ', count, ' Frames')
print('====================================================')
print('Movie to Frames HORUS - Done')
print('====================================================')
```

```
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0369.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0370.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0371.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0372.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0373.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0374.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0375.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0376.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0377.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0378.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0379.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0380.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0381.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0382.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0383.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0384.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp\dog-frame-0385.jpg
================================================
Generated :  386  Frames
================================================
Movie to Frames HORUS - Done
================================================
```

f2) Frames to Horus

```
# Utility Start Movie to HORUS (Part 2)
# Standard Tools
#============================================================
import imageio.v2 as imageio  # Use v2 explicitly to avoid warnings
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
```

```
all_frames = []  # To collect data frames
for file in os.listdir(sDataBaseDir):
    if file.endswith(".jpg"):
        f += 1
        sInputFileName = os.path.join(sDataBaseDir, file)
        print('Process : ', sInputFileName)
        InputData = imageio.imread(sInputFileName)
        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()
        all_frames.append(ProcessFrameData)  # Append frame to the list
if f > 0:
    ProcessData = pd.concat(all_frames, ignore_index=True)  # Use pd.concat to combine data frames
    sColumns = ['XAxis', 'YAxis', 'Red', 'Green', 'Blue', 'Alpha', 'FrameName']
    ProcessData.columns = sColumns
    ProcessData.index.names = ['ID']
    print('Rows: ', ProcessData.shape[0])
    print('Columns :', ProcessData.shape[1])
    print('=====================================================')
    # Output Agreement ============================================
    OutputData = ProcessData
    print('Storing File')
    sOutputFileName = 'C:/VKHCG/05-DS/9999-Data/HORUS-Movie-Frame.csv'
    OutputData.to_csv(sOutputFileName, index=False)
    print('=====================================================')
    print('Processed : ', f, ' frames')
    print('=====================================================')
    print('Movie to HORUS - Done')
    print('=====================================================')
```

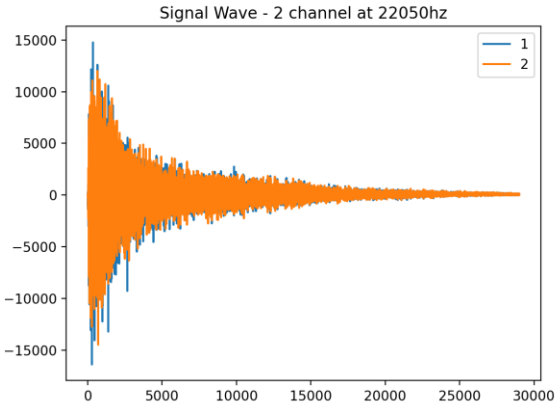


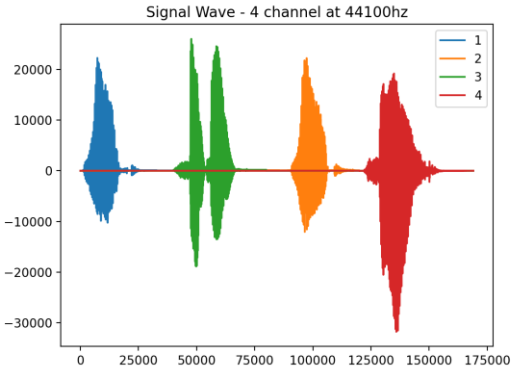
g) Audio to HORUS Format

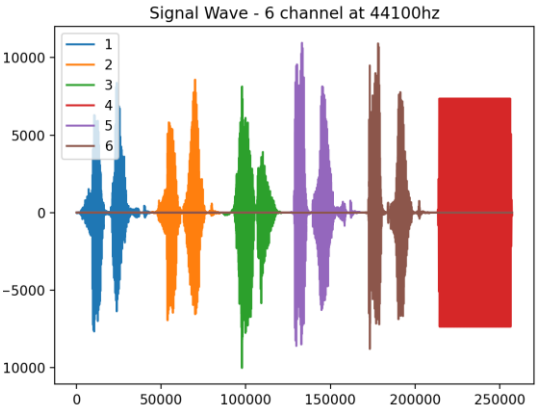
```
# Utility Start Audio to HORUS ================================
# Standard Tools
#=============================================================
from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#=============================================================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

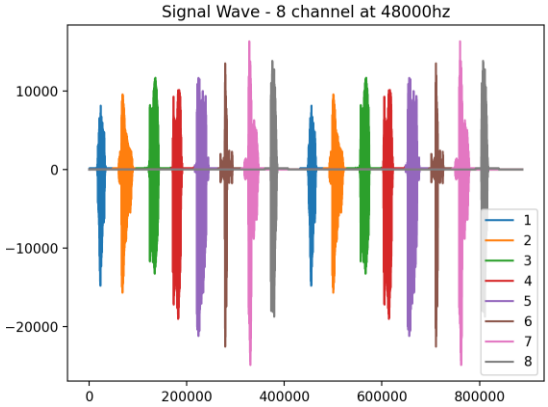
```
 plt.legend(sLegend)
 plt.show()
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
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')
```









Practical 2

Practical 2: Utilities and Auditing

a) 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,'<')
```

```
#1 Removing leading or lagging spaces from a data entry
>  Data Science with too many spaces is bad!!!  <
> Data Science with too many spaces is bad!!! <
PS C:\Users\krsna\Desktop\DS practical>
```

b) Removing nonprintable characters from a data entry

```
import string
import datetime as dt
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)
```

```
#2 Removing nonprintable characters from a data entry
Bad Data :  DataScience with funny characters is ►bad!!!
Clean Data :  D a t a S c i e n c e  w i t h  f u n n y  c h a r a c t e r s  i s  b a d ! ! !
PS C:\Users\krsna\Desktop\DS practical>
```

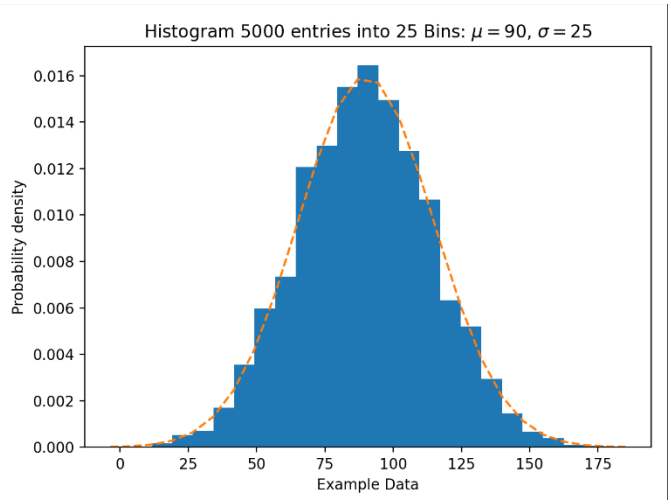
c) Reformatting data entry to match specific formatting criteria.

```
import string
import datetime as dt
# 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)
```

```
# 3 Reformatting data entry to match specific formatting criteria.
Bad Data :  2019-10-31
Good Data :  31 October 2019
```

d) Data Binning or Bucketing

```
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()
```



e) Averaging of Data

```
import pandas as pd
import sys
##################################################################
InputFileName = 'IP_DATA_CORE.csv'
OutputFileName = 'Retrieve_Router_Location.csv'
##################################################################
Base = 'C:/VKHCG'
print('################################')
print('Working Base :', Base, ' using ', sys.platform)
print('################################')
##################################################################
sFileName = Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :', sFileName)

# Read CSV file
IP_DATA_ALL = pd.read_csv(sFileName, header=0, low_memory=False,
                          usecols=['Country', 'Place Name', 'Latitude', 'Longitude'], encoding="latin-1")

# Rename column 'Place Name' to 'Place_Name'
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)

# Select specific columns
AllData = IP_DATA_ALL[['Country', 'Place_Name', 'Latitude']]
print(AllData)

# Calculate mean latitude grouped by 'Country' and 'Place_Name'
MeanData = AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
print(MeanData)
##################################################################
```

```
##############################
Working Base : C:/VKHCG  using  win32
##############################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
     Country Place_Name  Latitude
0         US   New York   40.7528
1         US   New York   40.7528
2         US   New York   40.7528
3         US   New York   40.7528
4         US   New York   40.7528
...      ...        ...       ...
3557      DE     Munich   48.0915
3558      DE     Munich   48.1833
3559      DE     Munich   48.1000
3560      DE     Munich   48.1480
3561      DE     Munich   48.1480

[3562 rows x 3 columns]
Country  Place_Name
DE       Munich        48.143223
GB       London        51.509406
US       New York      40.747044
Name: Latitude, dtype: float64
PS C:\Users\krsna\Desktop\DS practical> 
```

f) Outlier Detection

```
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)

###

```
c:\Users\krsna\Desktop\DS practical\p3\outlier detection.py:27: FutureWarning: Calling f
 in the future. Use float(ser.iloc[0]) instead
  LowerBound=float(MeanData-StdData)
Lower than  51.506176875621264
     Country Place_Name  Latitude
1915      GB     London   51.4739
Not Outliers
     Country Place_Name  Latitude
1917      GB     London   51.5085
1918      GB     London   51.5085
1922      GB     London   51.5085
1928      GB     London   51.5085
1929      GB     London   51.5085
...      ...        ...       ...
3432      GB     London   51.5092
3433      GB     London   51.5092
3434      GB     London   51.5092
3435      GB     London   51.5092
3437      GB     London   51.5085

[1485 rows x 3 columns]
PS C:\Users\krsna\Desktop\DS practical>
```

Audit

g) Logging

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

```
import sys
import os
import logging
import uuid
import shutil
import time
#################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
#################################################################
sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','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.')
```

```
Set up: C:/VKHCG/01-Vermeulen/01-Retrieve/Logging/Logging_dd7dad2a-0dd4-43f9-80ca-5d9fdb3
f26b4.log
root         : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-01-Retrieve-info: INFO     Practical Data Science logged informa
tion message.
Apllication-01-Vermeulen-01-Retrieve-warning: WARNING  Practical Data Science logged a wa
rning message.
Apllication-01-Vermeulen-01-Retrieve-error: ERROR    Practical Data Science logged an err
or message.
Set up: C:/VKHCG/01-Vermeulen/02-Assess/Logging/Logging_a590d97c-7013-456b-a510-43d820005
c3b.log
root         : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-02-Assess-info: INFO     Practical Data Science logged informati
on message.
Apllication-01-Vermeulen-02-Assess-warning: WARNING  Practical Data Science logged a warn
ing message.
Apllication-01-Vermeulen-02-Assess-error: ERROR    Practical Data Science logged an error
 message.
Set up: C:/VKHCG/01-Vermeulen/03-Process/Logging/Logging_54d1948e-df1c-445a-91e4-1ca4868a
6d4b.log
root         : INFO     Practical Data Science is fun!.
```

Practical 3

Practical 3: Retrieve Superstep

a) R Code Run in Online R Complier

```
# Create a sample dataset in base R
IP_DATA_ALL <- data.frame(
  Country = c("US", "GB", "US", "DE", "GB"),
  Place.Name = c("New York", "London", "Los Angeles", "Berlin", "Manchester"),
  Post.Code = c(10001, 12345, 90001, 10115, 54321),
  Latitude = c(40.7128, 51.5092, 34.0522, 52.5200, 53.4808),
  Longitude = c(-74.0060, -0.1180, -118.2437, 13.4050, -2.2426),
  stringsAsFactors = FALSE
)
# View the data
print(IP_DATA_ALL)
# Calculate mean latitude by Country and Place.Name
MeanData <- aggregate(Latitude ~ Country + Place.Name, data = IP_DATA_ALL, FUN = mean)
print(MeanData)
```

```
  Country  Place.Name Post.Code Latitude Longitude
1      US    New York     10001  40.7128  -74.0060
2      GB      London     12345  51.5092   -0.1180
3      US Los Angeles     90001  34.0522 -118.2437
4      DE      Berlin     10115  52.5200   13.4050
5      GB  Manchester     54321  53.4808   -2.2426
  Country  Place.Name Latitude
1      DE      Berlin  52.5200
2      GB      London  51.5092
3      US Los Angeles  34.0522
4      GB  Manchester  53.4808
5      US    New York  40.7128

=== Code Execution Successful ===
```

b) Program to retrieve different attributes of data.

```
import sys
import os
import pandas as pd
#####################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   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!! ############################################')
#######################################################################
```

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

c)Data Pattern - R code run in online compiler

```
############ SAMPLE DATASET (no CSV required) ############
IP_DATA_ALL <- data.frame(
  Country = c("India", "United States", "UK123", "Côte d'Ivoire", "Good Book 101"),
  stringsAsFactors = FALSE
)
############ Extract Unique Country Values ############
unique_countries <- unique(IP_DATA_ALL$Country)
############ Create Pattern Function ############
pattern_convert <- function(text) {
  # Replace letters with A
  text <- gsub("[A-Za-z]", "A", text)
  # Replace numbers with N
  text <- gsub("[0-9]", "N", text)
  # Replace spaces with b
  text <- gsub(" ", "b", text)
  # Replace special characters with u
  text <- gsub("[^ANb]", "u", text)
  return(text)
}
############ Apply Pattern Conversion ############
PatternCountry <- sapply(unique_countries, pattern_convert)
############ Show Results ############
output <- data.frame(Country = unique_countries, Pattern = PatternCountry)
print(output)
```

```
                   Country       Pattern
India                India         AAAAA
United States United States AAAAAAbAAAAAA
UK123                UK123         AANNN
Côte d'Ivoire Côte d'Ivoire AuAAbAuAAAAAA
Good Book 101 Good Book 101 AAAAbAAAAbNNN

=== Code Execution Successful ===
```

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!! ############################################')
######################################################################
```

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

e)**Vermeulen\Retrieve.**

```
import sys
import os
import pandas as pd
from math import radians, cos, sin, asin, sqrt
######################################################################
```

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

```
 haversine(
 row['Longitude_from'],
 row['Latitude_from'],
 row['Longitude_to'],
 row['Latitude_to'],
 'km')
 ,axis=1)
#####################################################################
IP_CROSS['DistanceBetweenMiles'] = IP_CROSS.apply(lambda row:
 haversine(
 row['Longitude_from'],
 row['Latitude_from'],
 row['Longitude_to'],
 row['Latitude_to'],
 'miles')
 ,axis=1)
print(IP_CROSS.shape)
sFileName2=sFileDir + '/Retrieve_IP_Routing.csv'
IP_CROSS.to_csv(sFileName2, index = False, encoding="latin-1")
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

```
ID,Country,Place Name,Post Code,Latitude,Longitude,First IP Number,Last IP Number
1,US,New York,10017,40.7528,-73.9725,204276480,204276735
2,US,New York,10017,40.7528,-73.9725,301984864,301985791
3,US,New York,10017,40.7528,-73.9725,404678736,404679039
4,US,New York,10017,40.7528,-73.9725,411592704,411592959
5,US,New York,10017,40.7528,-73.9725,416784384,416784639
6,US,New York,10017,40.7528,-73.9725,643347456,643348479
7,US,New York,10017,40.7528,-73.9725,643738112,643738623
8,US,New York,10017,40.7528,-73.9725,644459264,644459519
9,US,New York,10017,40.7528,-73.9725,644663808,644664319
10,US,New York,10017,40.7528,-73.9725,645591040,645591551
11,US,New York,10017,40.7528,-73.9725,789912128,789912191
12,US,New York,10017,40.7528,-73.9725,1064738304,1064738305
13,US,New York,10017,40.7528,-73.9725,1064769840,1064769847
```

f)Retrieve-Router-Location

```
import sys
import os
import pandas as pd
#####################################################################
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
#####################################################################
Base='C:/VKHCG'
#####################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
 usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
#####################################################################
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
#####################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
```

```
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])
sFileName2=sFileDir + '/' + OutputFileName
ROUTERLOC.to_csv(sFileName2, index = False, encoding="latin-1")
#################################################################
print('### Done!! ############################################')
```

```
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
Rows : 150
Columns : 4
### Done!! ############################################
```

g) Retrieve-DE-Billboard-Locations

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

```
ID,Country,PlaceName,Latitude,Longitude
1,DE,Lake,51.7833,8.5667
2,DE,Horb,48.4333,8.6833
3,DE,Hardenberg,51.1,7.7333
4,DE,Horn-bad Meinberg,51.9833,8.9667
5,DE,Winkel,51.55,13.3833
6,DE,Rohrdorf,47.7333,10.0833
7,DE,Sch<f6>nefeld,51.9833,12.8333
8,DE,Beuren,47.75,10.0167
9,DE,Marienfeld,50.8833,7.4333
10,DE,Borgholz,51.6167,9.2667
```

h) **Understanding Your Online Visitor Data**

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

```
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,
usecols=['Country','Latitude','Longitude'])
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA_ALL.rename(columns={'First IP Number': 'First_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Last IP Number': 'Last_IP_Number'}, inplace=True)
#################################################################
sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
visitordata = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
visitordata10=visitordata.head(10)
print('Rows :',visitordata.shape[0])
print('Columns :',visitordata.shape[1])
print('Export CSV')
sFileName2=sFileDir + '/' + OutputFileName + '.csv'
visitordata.to_csv(sFileName2, index = False)
print('Store All:',sFileName2)
sFileName3=sFileDir + '/' + OutputFileName + '_10.csv'
visitordata10.to_csv(sFileName3, index = False)
print('Store 10:',sFileName3)
for z in ['gzip', 'bz2', 'xz']:
   if z == 'gzip':
      sFileName4=sFileName2 + '.gz'
   else:
      sFileName4=sFileName2 + '.' + z
visitordata.to_csv(sFileName4, index = False, compression=z)
print('Store :',sFileName4)
#################################################################
print('Export JSON')
for sOrient in ['split','records','index', 'columns','values','table']:
   sFileName2=sFileDir + '/' + OutputFileName + '_' + sOrient + '.json'
visitordata.to_json(sFileName2,orient=sOrient,force_ascii=True)
print('Store All:',sFileName2)
sFileName3=sFileDir + '/' + OutputFileName + '_10_' + sOrient + '.json'
visitordata10.to_json(sFileName3,orient=sOrient,force_ascii=True)
print('Store 10:',sFileName3)
sFileName4=sFileName2 + '.gz'
file_in = open(sFileName2, 'rb')
file_out = gz.open(sFileName4, 'wb')
file_out.writelines(file_in)
file_in.close()
file_out.close()
print('Store GZIP All:',sFileName4)
sFileName5=sFileDir + '/' + OutputFileName + '_' + sOrient + '_UnGZip.json'
file_in = gz.open(sFileName4, 'rb')
file_out = open(sFileName5, 'wb')
file_out.writelines(file_in)
file_in.close()
file_out.close()
```

```
print('Store UnGZIP All:',sFileName5)
########################################################################
print('### Done!! ################################################')
########################################################################
```

```
PS C:\Users\krsna\Desktop\DS practical> & C:/Python312/python.exe "c:/Users/krsna/Desktop/DS practical/p4/6 online visitor d
##############################
Working Base : C:/VKHCG  using  win32
##############################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows : 1247502
Columns : 6
Export CSV
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10.csv
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows : 95257
Columns : 3
Export CSV
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10.csv
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.xz
Export JSON
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_table.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_table.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_table.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_table_UnGZip.json
### Done!! ############################################
PS C:\Users\krsna\Desktop\DS practical>
```

I)XML processing.

```
+ CompanyIn + '/00-RawData/' + InputFileName
print('Loading :', sFileName)
# Read CSV file
IP_DATA_ALL = pd.read_csv(sFileName, header=0, low_memory=False)
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name', 'First IP Number': 'First_IP_Number', 'Last IP Number':
'Last_IP_Number', 'Post Code': 'Post_Code'}, inplace=True)
########################################################################
import sys
import os
import pandas as pd
import xml.etree.ElementTree as ET
########################################################################
# Function to convert DataFrame to XML
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)
            text_value = str(data[schild][row]) if pd.notna(data[schild][row]) else 'n/a'
            child.text = text_value
        entry.append(child)
    result = ET.tostring(root)
    return result
# Function to convert XML to DataFrame
def xml2df(xml_data):
    try:
```

```
        root = ET.XML(xml_data)
    except ET.ParseError as e:
        print(f"XML ParseError: {e}")
        return pd.DataFrame()
    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)
######################################################################
InputFileName = 'IP_DATA_ALL.csv'
OutputFileName = 'Retrieve_Online_Visitor.xml'
CompanyIn = '01-Vermeulen'
CompanyOut = '02-Krennwallner'
######################################################################
# Set base directory based on OS
if sys.platform == 'linux':
    Base = os.path.expanduser('~') + '/VKHCG'
else:
    Base = 'C:/VKHCG'
######################################################################
print('##################################')
print('Working Base :', Base, ' using ', sys.platform)
print('##################################')
######################################################################
sFileName = Base + '/'
# Ensure output directory exists
sFileDir = Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
# Process data
visitordata = IP_DATA_ALL.head(10000)
print('Original Subset Data Frame')
print('Rows :', visitordata.shape[0])
print('Columns :', visitordata.shape[1])
print(visitordata)
# Convert DataFrame to XML and save
print('Export XML')
sXML = df2xml(visitordata)
sFileName = sFileDir + '/' + OutputFileName
with open(sFileName, 'wb') as file_out:
    file_out.write(sXML)
print('Store XML:', sFileName)
# Read XML back into DataFrame and remove duplicates
xml_data = open(sFileName).read()
unxmlrawdata = xml2df(xml_data)
print('Raw XML Data Frame')
print('Rows :', unxmlrawdata.shape[0])
```

```
print('Columns :', unxmlrawdata.shape[1])
print(unxmlrawdata)
unxmldata = unxmlrawdata.drop_duplicates()
print('Deduplicated XML Data Frame')
print('Rows :', unxmldata.shape[0])
print('Columns :', unxmldata.shape[1])
print(unxmldata)
#####################################################################
#print('### Done!! ############################################')
#####################################################################
```

```
Rows : 10000
Columns : 9
      Unnamed: 0     ID Country  ... Longitude First.IP.Number  Last.IP.Number
0              1      1      BW  ...   25.9119       692781056       692781567
1              2      2      BW  ...   25.9119       692781824       692783103
2              3      3      BW  ...   25.9119       692909056       692909311
3              4      4      BW  ...   25.9119       692909568       692910079
4              5      5      BW  ...   25.9119       693051392       693052415
...          ...    ...     ...  ...       ...             ...             ...
9995        9996   9996      US  ...  -83.3554      1144498560      1144498687
9996        9997   9997      US  ...  -83.3554      1144498816      1144499199
9997        9998   9998      US  ...  -83.3554      1144555136      1144555263
9998        9999   9999      US  ...  -83.3554      1144881664      1144881791
9999       10000  10000      US  ...  -83.3554      1171565568      1171566079

[10000 rows x 9 columns]
Export XML
Store XML: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.xml
XML ParseError: not well-formed (invalid token): line 1, column 22
Raw XML Data Frame
```

j) Retrieve-Incoterm-EXW

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

```
sFileName=sFileDir + '/' + OutputFileName
IncotermRule.to_csv(sFileName, index = False)
print('### Done!! ############################################')
```

```
##############################
Working Base : C:/VKHCG  using  win32
##############################
##########
Loading : C:/VKHCG/03-Hillman/00-RawData/Incoterm_2010.csv
Rows : 1
Columns : 9
##########
  Shipping_Term  Seller Carrier Port_From   Ship Port_To Terminal Named_Place  Buyer
0           EXW  Seller   Buyer     Buyer  Buyer   Buyer    Buyer       Buyer  Buyer
### Done!! ############################################
```

k) FCA—Free Carrier (Named Place of Delivery)

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

```
Loading : C:/VKHCG/03-Hillman/00-RawData/Incoterm_2010.csv
Rows : 1
Columns : 9
##########
  Shipping_Term  Seller Carrier Port_From   Ship Port_To Terminal Named_Place  Buyer
1           FCA  Seller  Seller     Buyer  Buyer   Buyer    Buyer       Buyer  Buyer
### Done!! ############################################
```

l)Person Base Data(Clark Ltd)

```
import sys
import os
import shutil
import zipfile
import pandas as pd
#################################################################
Base='C:/VKHCG'
#################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#################################################################
Company='04-Clark'
ZIPFiles=['Data_female-names','Data_male-names','Data_last-names']
for ZIPFile in ZIPFiles:
 InputZIPFile=Base+'/'+Company+'/00-RawData/' + ZIPFile + '.zip'
 OutputDir=Base+'/'+Company+'/01-Retrieve/01-EDS/02-Python/' + ZIPFile
 OutputFile=Base+'/'+Company+'/01-Retrieve/01-EDS/02-Python/Retrieve-'+ZIPFile+'.csv'
 zip_file = zipfile.ZipFile(InputZIPFile, 'r')
 zip_file.extractall(OutputDir)
 zip_file.close()
 t=0
 for dirname, dirnames, filenames in os.walk(OutputDir):
    for filename in filenames:
       sCSVFile = dirname + '/' + filename
       t=t+1
       if t==1:
          NameRawData=pd.read_csv(sCSVFile,header=None,low_memory=False)
          NameData=NameRawData
       else:
          NameRawData=pd.read_csv(sCSVFile,header=None,low_memory=False)
          NameData=NameData._append(NameRawData)
 NameData.rename(columns={0 : 'NameValues'},inplace=True)
 NameData.to_csv(OutputFile, index = False)
 shutil.rmtree(OutputDir)
 print('Process: ',InputZIPFile)
print('### Done!! ############################################')
```

```
Process:  C:/VKHCG/04-Clark/00-RawData/Data_female-names.zip
Process:  C:/VKHCG/04-Clark/00-RawData/Data_male-names.zip
Process:  C:/VKHCG/04-Clark/00-RawData/Data_last-names.zip
### Done!! ############################################
```

m)Retrieve-Route-Plan (Create a Delivery Route)

```
import os
import sys
import pandas as pd
from geopy.distance import geodesic
#####################################################################
# Input and Output Files
InputFileName = 'GB_Postcode_Warehouse.csv'
OutputFileName = 'Retrieve_GB_Warehouse.csv'
Company = '03-Hillman'
Base = 'C:/VKHCG'
print('##################################')
print('Working Base:', Base, 'using', sys.platform)
print('##################################')
#####################################################################
# Directory Setup
sFileDir = os.path.join(Base, Company, '01-Retrieve/01-EDS/02-Python')
os.makedirs(sFileDir, exist_ok=True)
#####################################################################
# Load Data
sFileName = os.path.join(Base, Company, '00-RawData', InputFileName)
print('############')
print('Loading:', sFileName)
Warehouse = pd.read_csv(sFileName, header=0, low_memory=False)
# Clean Data
WarehouseClean = Warehouse[Warehouse.latitude != 0]
WarehouseGood = WarehouseClean[WarehouseClean.longitude != 0]
WarehouseGood.drop_duplicates(subset='postcode', keep='first', inplace=True)
WarehouseGood.sort_values(by='postcode', ascending=True, inplace=True)
#####################################################################
# Save Cleaned Data
sFileName = os.path.join(sFileDir, OutputFileName)
WarehouseGood.to_csv(sFileName, index=False)
#####################################################################
# Process Route Information
WarehouseLoop = WarehouseGood.head(20)
for i in range(WarehouseLoop.shape[0]):
    print('Run:', i, ' =======>>>>>>>>>>>', WarehouseLoop.iloc[i]['postcode'])
# Create a working copy of WarehouseGood for processing
WarehouseHold = WarehouseGood.head(10000).copy()
# Add Transaction Type
WarehouseHold.loc[:, 'Transaction'] = 'WH-to-WH'
# Example Route Name
OutputLoopName = f"Retrieve_Route_WH-{WarehouseLoop.iloc[0]['postcode']}_Route.csv"
# Add Seller and Buyer Information
WarehouseHold.loc[:, 'Seller'] = f"WH-{WarehouseLoop.iloc[0]['postcode']}"
WarehouseHold.loc[:, 'Seller_Latitude'] = WarehouseLoop.iloc[0]['latitude']
WarehouseHold.loc[:, 'Seller_Longitude'] = WarehouseLoop.iloc[0]['longitude']
WarehouseHold.loc[:, 'Buyer'] = WarehouseHold['postcode'].apply(lambda x: f"WH-{x}")
WarehouseHold.loc[:, 'Buyer_Latitude'] = WarehouseHold['latitude']
```

```
WarehouseHold.loc[:, 'Buyer_Longitude'] = WarehouseHold['longitude']
# Calculate Distances
WarehouseHold.loc[:, 'Distance'] = WarehouseHold.apply(
   lambda row: round(
      geodesic(
         (WarehouseLoop.iloc[0]['latitude'], WarehouseLoop.iloc[0]['longitude']),
         (row['latitude'], row['longitude'])
      ).miles, 6),
   axis=1
)
# Drop Unnecessary Columns
WarehouseHold.drop(['id', 'postcode', 'latitude', 'longitude'], axis=1, inplace=True)
#####################################################################
# Save Processed Data
sFileLoopName = os.path.join(sFileDir, OutputLoopName)
WarehouseHold.to_csv(sFileLoopName, index=False)
#####################################################################
print('### Done!! ################################################')
```

```
Working Base: C:/VKHCG using win32
################################
###########
Loading: C:/VKHCG\03-Hillman\00-RawData\GB_Postcode_Warehouse.csv
Run: 0  =======>>>>>>>>>> AB10
Run: 1  =======>>>>>>>>>> AB11
Run: 2  =======>>>>>>>>>> AB12
Run: 3  =======>>>>>>>>>> AB13
Run: 4  =======>>>>>>>>>> AB14
Run: 5  =======>>>>>>>>>> AB15
Run: 6  =======>>>>>>>>>> AB16
Run: 7  =======>>>>>>>>>> AB21
Run: 8  =======>>>>>>>>>> AB22
Run: 9  =======>>>>>>>>>> AB23
Run: 10  =======>>>>>>>>>> AB24
Run: 11  =======>>>>>>>>>> AB25
Run: 12  =======>>>>>>>>>> AB30
Run: 13  =======>>>>>>>>>> AB31
Run: 14  =======>>>>>>>>>> AB32
Run: 15  =======>>>>>>>>>> AB33
Run: 16  =======>>>>>>>>>> AB34
Run: 17  =======>>>>>>>>>> AB35
Run: 18  =======>>>>>>>>>> AB36
Run: 19  =======>>>>>>>>>> AB37
### Done!! ############################################
```

Connecting to other Data Sources

n)Program to connect to different data sources.(SQLite**)**

```
import sqlite3 as sq
import pandas as pd
import os
#####################################################################
Base = 'C:/VKHCG'
sDatabaseDir = os.path.join(Base, '01-Vermeulen/00-RawData/SQLite')
sDatabaseName = os.path.join(sDatabaseDir, 'vermeulen.db')
# Ensure the directory exists
if not os.path.exists(sDatabaseDir):
   os.makedirs(sDatabaseDir)
# Connect to the SQLite database
conn = sq.connect(sDatabaseName)
```

```
#####################################################################
sFileName = os.path.join(Base, '01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv')
print('Loading :', sFileName)
IP_DATA_ALL_FIX = pd.read_csv(sFileName, header=0, low_memory=False)
IP_DATA_ALL_FIX.index.names = ['RowIDCSV']
# Save to SQLite
sTable = 'IP_DATA_ALL'
print('Storing :', sDatabaseName, ' Table:', sTable)
IP_DATA_ALL_FIX.to_sql(sTable, conn, if_exists="replace")
# Load from SQLite
print('Loading :', sDatabaseName, ' Table:', sTable)
TestData = pd.read_sql_query("SELECT * FROM IP_DATA_ALL;", conn)
# Display data
print('#################')
print('## Data Values')
print('#################')
print(TestData)
print('#################')
print('## Data Profile')
print('#################')
print('Rows :', TestData.shape[0])
print('Columns :', TestData.shape[1])
print('### Done!! ############################################')
```

```
Loading : C:/VKHCG\01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv
Storing : C:/VKHCG\01-Vermeulen/00-RawData/SQLite\vermeulen.db  Table: IP_DATA_ALL
Loading : C:/VKHCG\01-Vermeulen/00-RawData/SQLite\vermeulen.db  Table: IP_DATA_ALL
################
## Data Values
################
         RowIDCSV    RowID  Unnamed:.0  ...  Longitude First.IP.Number Last.IP.Number
0               0        0           1  ...    25.9119       692781056      692781567
1               1        1           2  ...    25.9119       692781824      692783103
2               2        2           3  ...    25.9119       692909056      692909311
3               3        3           4  ...    25.9119       692909568      692910079
4               4        4           5  ...    25.9119       693051392      693052415
...           ...      ...         ...  ...        ...              ...            ...
1247497   1247497  1247497     1247498  ...   -79.7611      1068157850     1068157850
1247498   1247498  1247498     1247499  ...     8.7668      1334409600     1334409607
1247499   1247499  1247499     1247500  ...    43.2583      1596886528     1596886783
1247500   1247500  1247500     1247501  ...   -80.4451      1742189568     1742190591
1247501   1247501  1247501     1247502  ...   139.5357      1905782573     1905782573

[1247502 rows x 11 columns]
################
## Data Profile
################
Rows : 1247502
Columns : 11
### Done!! ############################################
```

O) Global Post Codes - Load the global countries dataset, remove the unnecessary columns, and retrieve only the cleaned country list.
(R code run in online compiler)

```
# Sample data (acting as All_Countries.txt)
data <- data.frame(
  X1 = c("IN","US","UK"),
  X2 = c("India","USA","United Kingdom"),
  X6 = c(111,222,333),
  X7 = c("drop","drop","drop"),
  X8 = c("skip","skip","skip"),
  X9 = c("sample","sample","sample"),
  X12 = c("remove","remove","remove")
)
# Remove (skip) extra columns — logic same as original
clean_data <- subset(data, select = -c(X6, X7, X8, X9, X12))
# Display final cleaned data instead of saving to CSV
print("Cleaned Country Data:")
print(clean_data)
```

```
[1] "Cleaned Country Data:"
  X1             X2
1 IN          India
2 US            USA
3 UK United Kingdom

=== Code Execution Successful ===
```

p) Retrieve-Country-Currency (Microsoft Excel)

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

```
x.strip())
CurrencyData['CurrencyCode'] = CurrencyData['CurrencyCode'].map(lambda x:
x.strip())
print(CurrencyData)
print('~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~ ')
#####################################################################
sFileName=sFileDir + '/Retrieve-Country-Currency.csv'
CurrencyData.to_csv(sFileName, index = False)
```

Practical 4

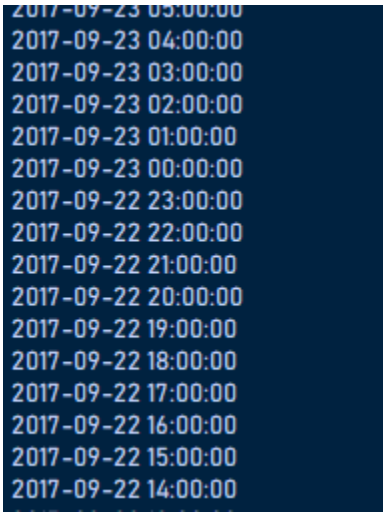
Practical 4: Processing Data

Build the time hub, links, and satellites

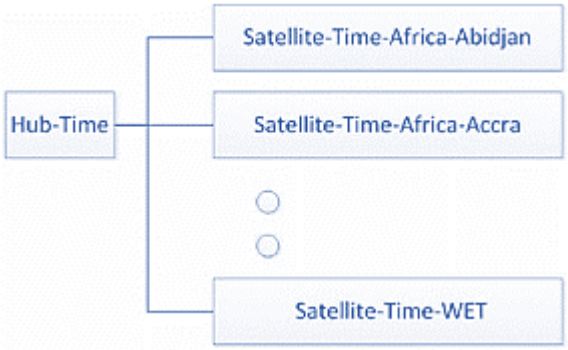
```
import sys
import os
from datetime import datetime
from datetime import timedelta
from pytz import timezone, all_timezones
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid

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_dict(TimeLine)
    else:
       TimeRow = pd.DataFrame.from_dict(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_dict(TimeZoneLine)
       else:
          TimeZoneRow = pd.DataFrame.from_dict(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.

```
################################################################
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!! ############################################')
################################################################
```

```
==== RESTART: C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p6\2.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: Hub-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.

```
import sys
import os
import pandas as pd
```

```
import sqlite3 as sq
from pandas.io import sql
import uuid

pd.options.mode.chained_assignment = None
#################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#################################################################
Company='03-Hillman'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
#################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
#################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('############')
print('Loading :',sFileName)
VehicleRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#################################################################
sTable='Process_Vehicles'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleRaw.to_sql(sTable, conn1, if_exists="replace")
#################################################################
VehicleRawKey=VehicleRaw[['Make','Model']].copy()
VehicleKey=VehicleRawKey.drop_duplicates()
#################################################################
VehicleKey['ObjectKey']=VehicleKey.apply(lambda row:
   str('('+ str(row['Make']).strip().replace(' ', '-').replace('/', '-').lower() +
    ')-(' + (str(row['Model']).strip().replace(' ', '-').replace(' ', '-').lower())
    +')')
     ,axis=1)
#################################################################
VehicleKey['ObjectType']=VehicleKey.apply(lambda row:
   'vehicle'
     ,axis=1)
#################################################################
```

```
VehicleKey['ObjectUUID']=VehicleKey.apply(lambda row:
   str(uuid.uuid4())
     ,axis=1)
#####################################################################
### Vehicle Hub
#####################################################################
#
VehicleHub=VehicleKey[['ObjectType','ObjectKey','ObjectUUID']].copy()
VehicleHub.index.name='ObjectHubID'
sTable = 'Hub-Object-Vehicle'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleHub.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
### Vehicle Satellite
#####################################################################
#
VehicleSatellite=VehicleKey[['ObjectType','ObjectKey','ObjectUUID','Make','Model']].copy()
VehicleSatellite.index.name='ObjectSatelliteID'
sTable = 'Satellite-Object-Make-Model'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleSatellite.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
### Vehicle Dimension
#####################################################################
sView='Dim-Object'
print('Storing :',sDatabaseName,' View:',sView)
sSQL="CREATE VIEW IF NOT EXISTS [" + sView + "] AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   H.ObjectType,"
sSQL=sSQL+ "   H.ObjectKey AS VehicleKey,"
sSQL=sSQL+ "   TRIM(S.Make) AS VehicleMake,"
sSQL=sSQL+ "   TRIM(S.Model) AS VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   [Hub-Object-Vehicle] AS H"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ "   [Satellite-Object-Make-Model] AS S"
sSQL=sSQL+ " ON"
sSQL=sSQL+ "    H.ObjectType=S.ObjectType"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " H.ObjectUUID=S.ObjectUUID;"
sql.execute(sSQL,conn2)

print('#################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " VehicleMake,"
sSQL=sSQL+ " VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " [" + sView + "]"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " VehicleMake"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " VehicleMake;"
DimObjectData=pd.read_sql_query(sSQL, conn2)
```

```
DimObjectData.index.name='ObjectDimID'
DimObjectData.sort_values(['VehicleMake','VehicleModel'],inplace=True, ascending=True)
print('#################')
print(DimObjectData)
#########################################################################
print('#################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#################')
#########################################################################
conn1.close()
conn2.close()
#########################################################################
#print('### Done!! ############################################')
#########################################################################
```

```
==== RESTART: C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p6\3.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

Warning (from warnings module):
  File "C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p6\3.py", line 100
    sql.execute(sSQL,conn2)
FutureWarning: `pandas.io.sql.execute` is deprecated and will be removed in the future version.
################
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

[3776 rows x 2 columns]
################
Vacuum Databases
#################
```

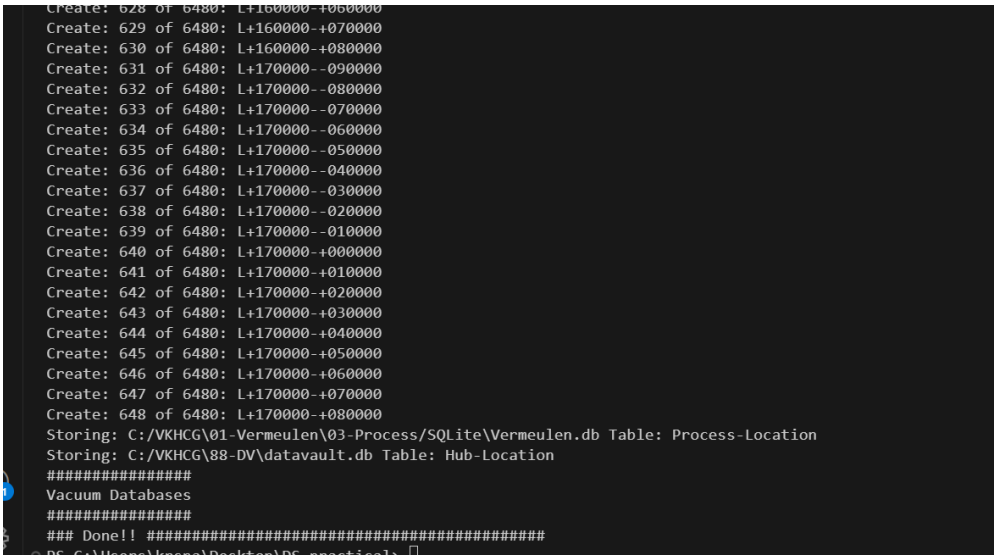
Human-Environment Interaction

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.

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

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



Forecasting

Forecasting is the ability to project a possible future, by looking at historical data. The data vault enables these types of investigations, owing to the complete history it collects as it processes 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.

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

```
print('#################################')
######################################################################
Company='04-Clark'
sInputFileName='00-RawData/VKHCG_Shares.csv'
sOutputFileName='Shares.csv'
######################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
######################################################################
sFileDir1=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir1):
   os.makedirs(sFileDir1)
######################################################################
sFileDir2=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir2):
   os.makedirs(sFileDir2)
######################################################################
sFileDir3=Base + '/' + Company + '/03-Process/01-EDS/02-Python'
if not os.path.exists(sFileDir3):
   os.makedirs(sFileDir3)
######################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
######################################################################
### Import Share Names Data
######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Rows   :',RawData.shape[0])
print('Columns:',RawData.shape[1])
print('#################')
######################################################################
sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
print('#################################')
print('Storing :', sFileName)
print('#################################')
RawData.to_csv(sFileName, index = False)
print('#################################')
######################################################################
sFileName=sFileDir2 + '/Assess_' + sOutputFileName
print('#################################')
print('Storing :', sFileName)
print('#################################')
RawData.to_csv(sFileName, index = False)
print('#################################')
######################################################################
sFileName=sFileDir3 + '/Process_' + sOutputFileName
print('#################################')
print('Storing :', sFileName)
```

```
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
#####################################################################
### Import Shares Data Details
#####################################################################
nShares=RawData.shape[0]
#nShares=6
for sShare in range(nShares):
   sShareName=str(RawData['Shares'][sShare])
   ShareData = quandl.get(sShareName)
   UnitsOwn=RawData['Units'][sShare]
   ShareData['UnitsOwn']=ShareData.apply(lambda row:(UnitsOwn),axis=1)
   ShareData['ShareCode']=ShareData.apply(lambda row:(sShareName),axis=1)
   print('################')
   print('Share  :',sShareName)
   print('Rows   :',ShareData.shape[0])
   print('Columns:',ShareData.shape[1])
   print('################')
   #####################################################################
   print('################')
   sTable=str(RawData['sTable'][sShare])
   print('Storing :',sDatabaseName,' Table:',sTable)
   ShareData.to_sql(sTable, conn, if_exists="replace")
   print('################')
   #####################################################################
   sOutputFileName = sTable.replace("/","-") + '.csv'
   sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
   print('################################')
   print('Storing :', sFileName)
   print('################################')
   ShareData.to_csv(sFileName, index = False)
   print('################################')
   #####################################################################
   sOutputFileName = sTable.replace("/","-") + '.csv'
   sFileName=sFileDir2 + '/Assess_' + sOutputFileName
   print('################################')
   print('Storing :', sFileName)
   print('################################')
   ShareData.to_csv(sFileName, index = False)
   print('################################')
   #####################################################################
   sOutputFileName = sTable.replace("/","-") + '.csv'
   sFileName=sFileDir3 + '/Process_' + sOutputFileName
   print('################################')
   print('Storing :', sFileName)
   print('################################')
   ShareData.to_csv(sFileName, index = False)
   print('################################')
print('### Done!! ############################################')
#####################################################################
```

```
################################
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\Retrieve_Shares.csv
################################
################################
Storing: C:/VKHCG\04-Clark\03-Process/01-EDS/02-Python\Retrieve_Shares.csv
################################
Fetching data for: WIKI/GOOGL
Error processing share 1/10: (Status 403) Something went wrong. Please try again. If you continue to have pro
blems, please contact us at connect@quandl.com.
Fetching data for: WIKI/MSFT
```

Practical 5

Practical 5: Transforming Data

a)

```
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.info(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")
#####################################################################
```

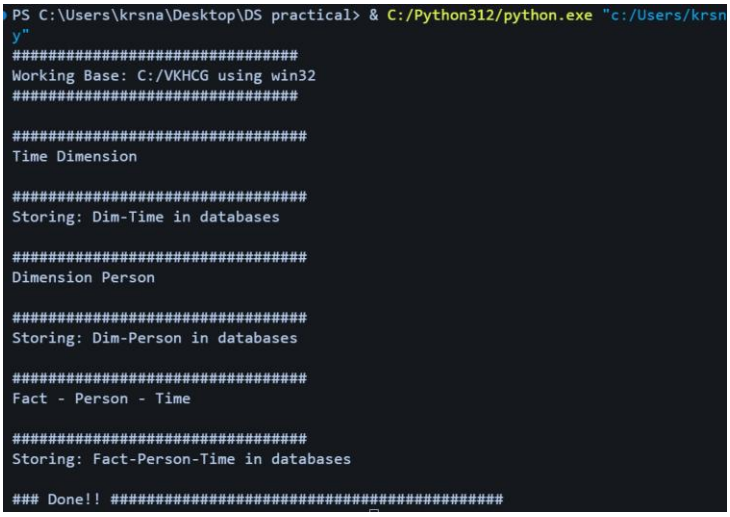
```
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson_is_Born.py
Working Base : C:/VKHCG  using  win32
Time Category
UTC Time
1960-12-20 10:15:00 (UTC) (+0000)
################################
Birth Date in Reykjavik :
1960-12-20 09:15:00 (-01) (-0100)
################################
################################
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
################################
```

b)

```
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")
#####################################################################
```



Building a Data Warehouse

```
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(dict(TimeLine))
        else:
            TimeRow = pd.DataFrame(dict(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")
#####################################################################
```

```
==== RESTART: C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p7\c.py ===
################################
Working Base : C:/VKHCG using win32
################################
      DateTimeValue
0   2018-01-01 00:00:00
1   2017-12-31 23:00:00
2   2017-12-31 22:00:00
3   2017-12-31 21:00:00
4   2017-12-31 20:00:00
..                  ...
995 2017-11-20 13:00:00
996 2017-11-20 12:00:00
997 2017-11-20 11:00:00
998 2017-11-20 10:00:00
999 2017-11-20 09:00:00

[1000 rows x 1 columns]

################################
Time Dimension

################################
1 3000
2 3000
3 3000
4 3000
5 3000
```

.

.

.

```
994 1000
995 1000
996 1000
997 1000
998 1000
999 1000
1000 1000
                PersonID  ...                          BirthDate
0  5bd65664-981c-435b-bd2a-d827e991ebe0  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
0  eedd7ac7-fb3b-42cc-ac4b-72d76743d60c  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
0  2def30a5-4ab9-463d-b7b8-05f21b5ebc81  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
0  da18b495-46cb-45f0-ab37-a96cad037b26  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
0  3b0a5e5e-4c25-421e-911a-9818aca4036a  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
..                                     ...  ...                          ...
0  35b1a0eb-8f3a-4624-81bf-ef8bcb765ed6  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
0  5bbe131c-da01-448d-80b0-77abdfd3803f  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
0  9b929776-b260-4ca5-b0b1-2c02ff67494a  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
0  d846cfe7-e33c-46b3-9646-39b8fe9b95b1  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...
0  993b7ee5-e18a-4e12-98b1-213c707a2ccc  ...  0   1961-07-22 20:00:00\n1    1987-12-10 ...

[1000 rows x 6 columns]

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

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

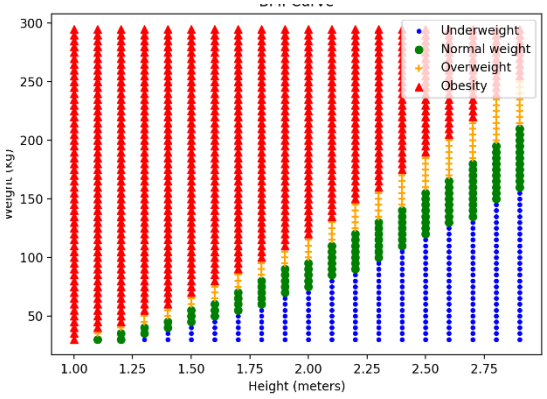
Simple Linear Regression

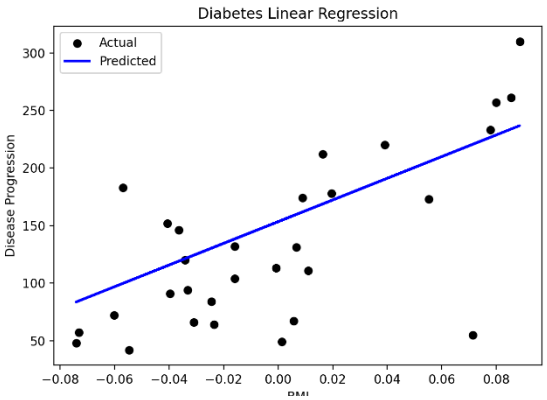
```
import sys
import os
import pandas as pd
import sqlite3 as sq
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
################################################################
Base='C:/VKHCG'
print('##################################')
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_dict(PersonLine)
    else:
       PersonRow = pd.DataFrame.from_dict(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()
```





Practical 6

Practical 6: Organizing Data

Horizontal Style

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
   os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####################################################################
print('##################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
print('##################################')
sSQL="SELECT PersonID,\
     Height,\
     Weight,\
     bmi,\
     Indicator\
  FROM [Dim-BMI]\
  WHERE \
  Height > 1.5 \
  and Indicator = 1\
  ORDER BY  \
     Height,\
     Weight;"
```

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

```
= RESTART: C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p8\Horizontal Style.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-Horizontal

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

Vertical Style

```
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('Vertical Data Set (Rows):', PersonFrame2.shape[0])
print('Vertical Data Set (Columns):', PersonFrame2.shape[1])
print('#################################')
#####################################################################
```

```
= RESTART: C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p8\Vertical Style.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
################################
Vertical Data Set (Rows): 1080
Vertical Data Set (Columns): 3
################################
```

Island Style

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

```
= RESTART: C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p8\ISLAND style.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

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

```
= RESTART: C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p8\Secure Vault Style.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

```
######################################################################
# -*- 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!! ############################################')
######################################################################
```

```
= RESTART: C:\Users\hirna\OneDrive\Desktop\Mscit\Data Science\dsp\p8\Associate rule minnin
g.py
###################################
Working Base : C:/VKHCG  using  win32
###################################
(541909, 8)
```

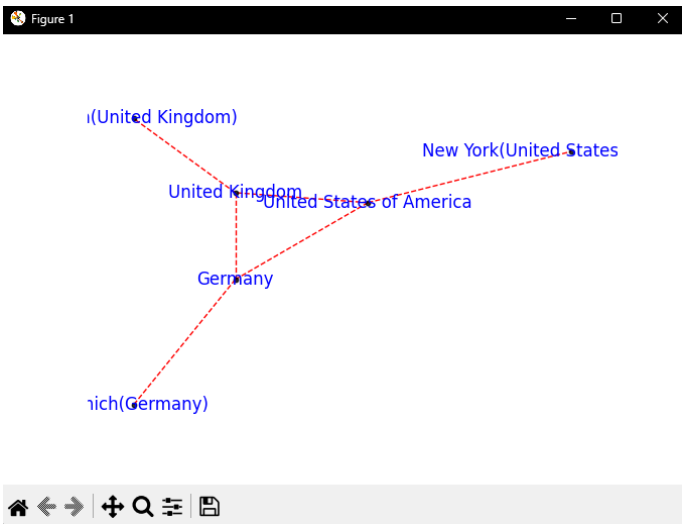
```
                antecedents  ... zhangs_metric
0   (ALARM CLOCK BAKELIKE PINK)  ...     0.964734
1 (ALARM CLOCK BAKELIKE GREEN)  ...     0.959283
2    (ALARM CLOCK BAKELIKE RED)  ...     0.976465
3 (ALARM CLOCK BAKELIKE GREEN)  ...     0.979224
4   (ALARM CLOCK BAKELIKE PINK)  ...     0.968652

[5 rows x 10 columns]
ALARM CLOCK BAKELIKE GREEN
340.0
ALARM CLOCK BAKELIKE RED
316.0
```

```
                antecedents  ... zhangs_metric
1   (PLASTERS IN TIN CIRCUS PARADE)  ...     0.864580
7       (PLASTERS IN TIN SPACEBOY)  ...     0.849877
10   (RED RETROSPOT CHARLOTTE BAG)  ...     0.913551

[3 rows x 10 columns]
### Done!! ############################################
```

Create a Network Routing Diagram

```
######################################################################
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]):
```

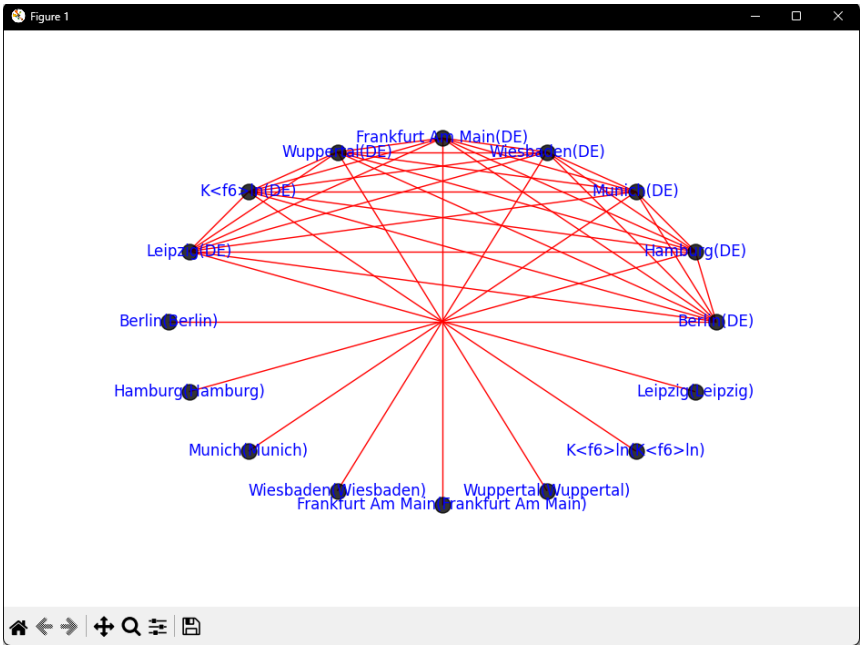
```
 for j in range(CompanyData.shape[0]):
    Node0=CompanyData['Company_Country_Name'][i]
    Node1=CompanyData['Company_Country_Name'][j]
    if Node0 != Node1:
       G.add_edge(Node0,Node1)
for i in range(CompanyData.shape[0]):
 Node0=CompanyData['Company_Country_Name'][i]
 Node1=CompanyData['Company_Place_Name'][i]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

```
#####################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
#####################################################################
pd.options.mode.chained_assignment = None
#####################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/VKHCG'
#####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-DE-Billboard-Visitor.csv'
#####################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Billboards.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Billboards.png'
Company='02-Krennwallner'
#####################################################################
#####################################################################
### Import Company Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
BillboardDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('##################################')
#####################################################################
print(BillboardDataRaw.head())
print(BillboardDataRaw.shape)
BillboardData=BillboardDataRaw
sSample=list(np.random.choice(BillboardData.shape[0],20))
#####################################################################
G=nx.Graph()
for i in sSample:
   for j in sSample:
      Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['BillboardCountry'][i] + ')'
      Node1=BillboardData['BillboardPlaceName'][j] + '('+ BillboardData['BillboardCountry'][i] + ')'
      if Node0 != Node1:
         G.add_edge(Node0,Node1)
```

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

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



Create a Delivery Route

```
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
#####################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/VKHCG'
#####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
sInputFileName='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, sep='|', encoding="utf-8")
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('#################################')
```

```
PS C:\Users\krsna\Desktop\DS practical> & C:/Python312/python.exe "c:/Users/krsna/Desktop
/DS practical/p8/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
################################
```

Simple Forex Trading Planner

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import re
#######################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
#######################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#######################################################################
sInputFileName='03-Process/01-EDS/02-Python/Process_ExchangeRates.csv'
#######################################################################
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Forex.csv'
Company='04-Clark'
#######################################################################
sDatabaseName=Base + '/' + Company + '/05-Organise/SQLite/clark.db'
conn = sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
#######################################################################
#######################################################################
### Import Forex Data
#######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('#################################')
#######################################################################
ForexDataRaw.index.names = ['RowID']
```

```
sTable='Forex_All'
print('Storing :',sDatabaseName,' Table:',sTable)
ForexDataRaw.to_sql(sTable, conn, if_exists="replace")
#####################################################################
sSQL="SELECT 1 as Bag\
     , CAST(min(Date) AS VARCHAR(10)) as Date \
     ,CAST(1000000.0000000 as NUMERIC(12,4)) as Money \
     ,'USD' as Currency \
     FROM Forex_All \
     ;"
sSQL=re.sub("\s\s+", " ", sSQL)
nMoney=pd.read_sql_query(sSQL, conn)

#####################################################################
nMoney.index.names = ['RowID']
sTable='MoneyData'
print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
#####################################################################
sTable='TransactionData'
print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
#####################################################################
ForexDay=pd.read_sql_query("SELECT Date FROM Forex_All GROUP BY Date;", conn)
#####################################################################
t=0
for i in range(ForexDay.shape[0]):
   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)
#####################################################################
```

```
Trade : 518 20010105 USD 1000282.87
Trade : 519 20010108 GBP 666504.16
Trade : 520 20010109 USD 992367.06
Trade : 521 20010110 GBP 665092.59
Trade : 522 20010111 USD 997429.41
Trade : 523 20010112 GBP 669620.5
Trade : 524 20010115 USD 988287.74
Trade : 525 20010116 GBP 672753.88
Trade : 526 20010117 USD 992961.9
Trade : 527 20010118 GBP 674187.76
Trade : 528 20010119 USD 993317.39
Trade : 529 20010122 GBP 683239.84
Trade : 530 20010123 USD 1006142.32
Trade : 531 20010124 GBP 686458.84
Trade : 532 20010125 USD 995931.56
Trade : 533 20010126 GBP 681546.14
Trade : 534 20010129 USD 993885.42
Trade : 535 20010130 GBP 680601.32
Trade : 536 20010131 USD 993376.48
Trade : 537 20010201 GBP 672847.28
Trade : 538 20010202 USD 994574.85
Trade : 539 20010205 GBP 673989.34
Trade : 540 20010206 USD 986950.77
Trade : 541 20010207 GBP 675677.61
Trade : 542 20010208 USD 980290.73
Trade : 543 20010209 GBP 678033.54
Trade : 544 20010212 USD 984303.77
Trade : 545 20010213 GBP 676649.15
Trade : 546 20010214 USD 984821.74
Trade : 547 20010215 GBP 680274.56
Trade : 548 20010216 USD 986528.02
```

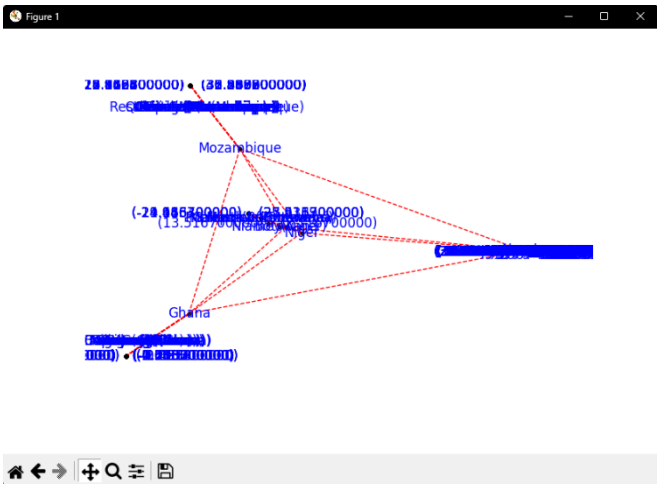
Practical 7

Practical 7: Generating Data

1) Report Superstep

```
######################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
######################################################################
pd.options.mode.chained_assignment = None
######################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/VKHCG'
######################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
######################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv'
######################################################################
sOutputFileName1='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml'
sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png'
Company='01-Vermeulen'
######################################################################
######################################################################
### Import Country Data
######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
CustomerDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
CustomerData=CustomerDataRaw.head(100)
print('Loaded Country:',CustomerData.columns.values)
print('##################################')
######################################################################
print(CustomerData.head())
print(CustomerData.shape)
######################################################################
G=nx.Graph()
for i in range(CustomerData.shape[0]):
   for j in range(CustomerData.shape[0]):
      Node0=CustomerData['Customer_Country_Name'][i]
      Node1=CustomerData['Customer_Country_Name'][j]
      if Node0 != Node1:
         G.add_edge(Node0,Node1)
for i in range(CustomerData.shape[0]):
   Node0=CustomerData['Customer_Country_Name'][i]
```

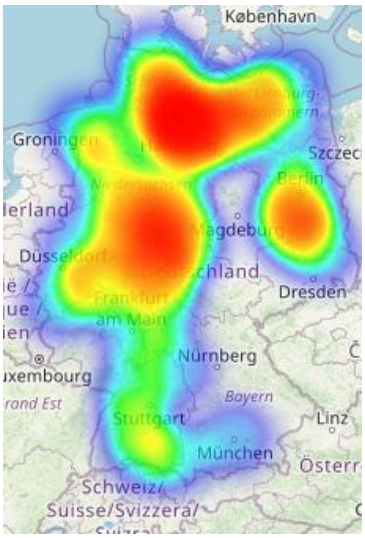
```
    Node1=CustomerData['Customer_Place_Name'][i] + '('+ CustomerData['Customer_Country_Name'][i] + ')'
    Node2='('+ "{:.9f}".format(CustomerData['Customer_Latitude'][i]) + ')\
    ('+ "{:.9f}".format(CustomerData['Customer_Longitude'][i]) + ')'
    if Node0 != Node1:
        G.add_edge(Node0,Node1)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

print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
#####################################################################
sFileName=Base + '/' + Company + '/' + sOutputFileName1
print('##################################')
print('Storing :',sFileName)
print('##################################')
nx.write_gml(G, sFileName)
#####################################################################
sFileName=Base + '/' + Company + '/' + sOutputFileName2
print('##################################')
print('Storing Graph Image:',sFileName)
print('##################################')

plt.figure(figsize=(25, 25))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
#####################################################################
print('##################################')
print('### Done!! ####################')
print('##################################')
#####################################################################
```



2)

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
from folium.plugins import FastMarkerCluster, HeatMap
from folium import Marker, Map
import webbrowser
#####################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
sFileName=Base+'/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
df = pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
df.fillna(value=0, inplace=True)
print(df.shape)
#####################################################################
t=0
for i in range(df.shape[0]):
   try:
      sLongitude=df["Longitude"][i]
      sLongitude=float(sLongitude)
   except Exception:
      sLongitude=float(0.0)
         try:
      sLatitude=df["Latitude"][i]
      sLatitude=float(sLatitude)
   except Exception:
      sLatitude=float(0.0)
      try:
      sDescription=df["Place_Name"][i] + ' (' + df["Country"][i]+')'
   except Exception:
      sDescription='VKHCG'
   if sLongitude != 0.0 and sLatitude != 0.0:
      DataClusterList=list([sLatitude, sLongitude])
      DataPointList=list([sLatitude, sLongitude, sDescription])
      t+=1
      if t==1:
         DataCluster=[DataClusterList]
         DataPoint=[DataPointList]
      else:
         DataCluster.append(DataClusterList)
         DataPoint.append(DataPointList)
```

```
data=DataCluster
pins=pd.DataFrame(DataPoint)
pins.columns = [ 'Latitude','Longitude','Description']
#####################################################################
stops_map1 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
marker_cluster = FastMarkerCluster(data).add_to(stops_map1)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard1.html'
stops_map1.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
#####################################################################
stops_map2 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
for name, row in pins.iloc[:100].iterrows():
   Marker([row["Latitude"],row["Longitude"]], popup=row["Description"]).add_to(stops_map2)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html'
stops_map2.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
#####################################################################
stops_heatmap = Map(location=[48.1459806, 11.4985484], zoom_start=5)
stops_heatmap.add_child(HeatMap([[row["Latitude"], row["Longitude"]] for name, row in
pins.iloc[:100].iterrows()]))
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard_heatmap.html'
stops_heatmap.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
#####################################################################
print('### Done!! ##############################################')
#####################################################################
```

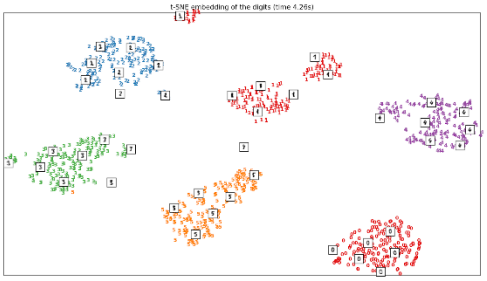


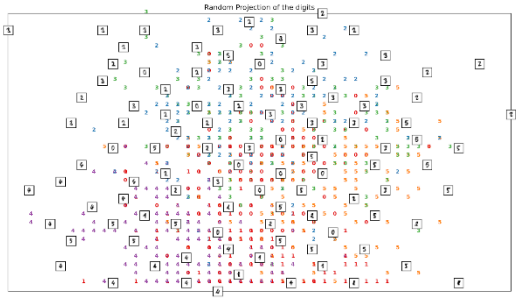
3)

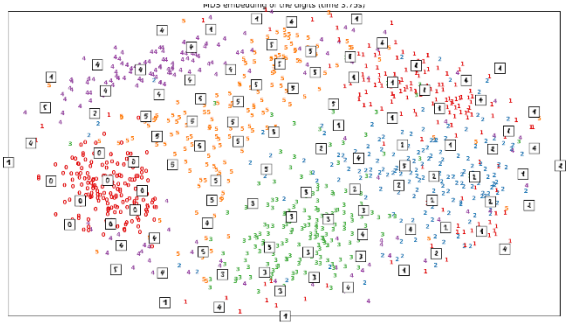
```
from time import time
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import offsetbox
from sklearn import (manifold, datasets, decomposition, ensemble, discriminant_analysis, random_projection)
digits = datasets.load_digits(n_class=6)
X = digits.data
y = digits.target
n_samples, n_features = X.shape
n_neighbors = 30
def plot_embedding(X, title=None):
 x_min, x_max = np.min(X, 0), np.max(X, 0)
 X = (X - x_min) / (x_max - x_min)
 plt.figure(figsize=(10, 10))
 ax = plt.subplot(111)
 for i in range(X.shape[0]):
    plt.text(X[i, 0], X[i, 1], str(digits.target[i]),
       color=plt.cm.Set1(y[i] / 10.),
       fontdict={'weight': 'bold', 'size': 9})
 if hasattr(offsetbox, 'AnnotationBbox'):
 # only print thumbnails with matplotlib > 1.0
    shown_images = np.array([[1., 1.]]) # just something big
    for i in range(digits.data.shape[0]):
       dist = np.sum((X[i] - shown_images) ** 2, 1)
       if np.min(dist) < 4e-3:
 # don't show points that are too close
          continue
       shown_images = np.r_[shown_images, [X[i]]]
       imagebox = offsetbox.AnnotationBbox(offsetbox.OffsetImage(digits.images[i],
cmap=plt.cm.gray_r),X[i])
       ax.add_artist(imagebox)
    plt.xticks([]), plt.yticks([])
    if title is not None:
       plt.title(title)
n_img_per_row = 20
img = np.zeros((10 * n_img_per_row, 10 * n_img_per_row))
for i in range(n_img_per_row):
    ix = 10 * i + 1
    for j in range(n_img_per_row):
       iy = 10 * j + 1
       img[ix:ix + 8, iy:iy + 8] = X[i * n_img_per_row + j].reshape((8, 8))
plt.figure(figsize=(10, 10))
plt.imshow(img, cmap=plt.cm.binary)
plt.xticks([])
plt.yticks([])
plt.title('A selection from the 64-dimensional digits dataset')
print("Computing random projection")
rp = random_projection.SparseRandomProjection(n_components=2, random_state=42)
X_projected = rp.fit_transform(X)
```

```
plot_embedding(X_projected, "Random Projection of the digits")
print("Computing PCA projection")
t0 = time()
X_pca = decomposition.TruncatedSVD(n_components=2).fit_transform(X)
plot_embedding(X_pca,"Principal Components projection of the digits (time %.2fs)" %(time() - t0))
print("Computing Linear Discriminant Analysis projection")
X2 = X.copy()
X2.flat[::X.shape[1] + 1] += 0.01 # Make X invertible
t0 = time()
X_lda = discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_transform(X2, y)
plot_embedding(X_lda,"Linear Discriminant projection of the digits (time %.2fs)" %(time() - t0))
print("Computing Isomap embedding")
t0 = time()
X_iso = manifold.Isomap(n_neighbors=5, n_components=2).fit_transform(X)
print("Done.")
plot_embedding(X_iso,"Isomap projection of the digits (time %.2fs)" %(time() - t0))
print("Computing LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors=5, n_components=2,method='standard')
t0 = time()
X_lle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_lle,"Locally Linear Embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing modified LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors=5, n_components=2,
method='modified')
t0 = time()
X_mlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_mlle,"Modified Locally Linear Embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing Hessian LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors=10, n_components=2,method='hessian')
t0 = time()
X_hlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_hlle,"Hessian Locally Linear Embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing LTSA embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors=10, n_components=2,method='ltsa')
t0 = time()
X_ltsa = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_ltsa,"Local Tangent Space Alignment of the digits (time %.2fs)" %(time() - t0))
print("Computing MDS embedding")
clf = manifold.MDS(n_components=2, n_init=1, max_iter=100)
t0 = time()
X_mds = clf.fit_transform(X)
print("Done. Stress: %f" % clf.stress_)
plot_embedding(X_mds,"MDS embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing Totally Random Trees embedding")
hasher = ensemble.RandomTreesEmbedding(n_estimators=200, random_state=0,
max_depth=5)
```

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







4)

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import re
from openpyxl import load_workbook
#####################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
```

```
else:
   Base='C:/VKHCG'
#################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#################################################################
sInputTemplateName='00-RawData/Balance-Sheet-Template.xlsx'
#################################################################
sOutputFileName='06-Report/01-EDS/02-Python/Report-Balance-Sheet'
Company='04-Clark'
#################################################################
sDatabaseName=Base + '/' + Company + '/06-Report/SQLite/clark.db'
conn = sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
#################################################################
### Import Balance Sheet Data
#################################################################
for y in range(1,13):
   sInputFileName='00-RawData/BalanceSheets' + str(y).zfill(2) + '.csv'
   sFileName=Base + '/' + Company + '/' + sInputFileName
   print('#################################')
   print('Loading :',sFileName)
   print('#################################')
   ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
   print('#################################')
   #################################################################
   ForexDataRaw.index.names = ['RowID']
   sTable='BalanceSheets'
   print('Storing :',sDatabaseName,' Table:',sTable)
   if y == 1:
      print('Load Data')
      ForexDataRaw.to_sql(sTable, conn, if_exists="replace")
   else:
      print('Append Data')
      ForexDataRaw.to_sql(sTable, conn, if_exists="append")
#################################################################
sSQL="SELECT \
         Year, \
         Quarter, \
         Country, \
         Company, \
         CAST(Year AS INT) || 'Q' || CAST(Quarter AS INT) AS sDate, \
         Company || ' (' || Country || ')' AS sCompanyName , \
         CAST(Year AS INT) || 'Q' || CAST(Quarter AS INT) || '-' ||\
         Company || '-' || Country AS sCompanyFile \
      FROM BalanceSheets \
      GROUP BY \
         Year, \
         Quarter, \
```

```
        Country, \
        Company \
      HAVING Year is not null \
     ;"
sSQL=re.sub("\s\s+", " ", sSQL)
sDatesRaw=pd.read_sql_query(sSQL, conn)
print(sDatesRaw.shape)
sDates=sDatesRaw.head(5)
#####################################################################
## Loop Dates
#####################################################################
for i in range(sDates.shape[0]):
   sFileName=Base + '/' + Company + '/' + sInputTemplateName
   wb = load_workbook(sFileName)
   ws=wb.get_sheet_by_name("Balance-Sheet")
   sYear=sDates['sDate'][i]
   sCompany=sDates['sCompanyName'][i]
   sCompanyFile=sDates['sCompanyFile'][i]
   sCompanyFile=re.sub("\s+", "", sCompanyFile)
    ws['D3'] = sYear
   ws['D5'] = sCompany
    sFields = pd.DataFrame(
         [
        ['Cash','D16', 1],
        ['Accounts_Receivable','D17', 1],
        ['Doubtful_Accounts','D18', 1],
        ['Inventory','D19', 1],
        ['Temporary_Investment','D20', 1],
        ['Prepaid_Expenses','D21', 1],
        ['Long_Term_Investments','D24', 1],
        ['Land','D25', 1],
        ['Buildings','D26', 1],
        ['Depreciation_Buildings','D27', -1],
        ['Plant_Equipment','D28', 1],
        ['Depreciation_Plant_Equipment','D29', -1],
        ['Furniture_Fixtures','D30', 1],
        ['Depreciation_Furniture_Fixtures','D31', -1],
        ['Accounts_Payable','H16', 1],
        ['Short_Term_Notes','H17', 1],
        ['Current_Long_Term_Notes','H18', 1],
        ['Interest_Payable','H19', 1],
        ['Taxes_Payable','H20', 1],
        ['Accrued_Payroll','H21', 1],
        ['Mortgage','H24', 1],
        ['Other_Long_Term_Liabilities','H25', 1],
        ['Capital_Stock','H30', 1]
        ]
         )

   nYear=str(int(sDates['Year'][i]))
```

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

sFileName=Base + '/' + Company + '/' + sOutputFileName + \
'-' + sCompanyFile + '.xlsx'
print(sFileName)
for j in range(sFields.shape[0]):
   sSumField=sFields[0][j]
   sCellField=sFields[1][j]
   nSumSign=sFields[2][j]
   sSQL="SELECT  \
         Year, \
         Quarter, \
         Country, \
         Company, \
         SUM(" + sSumField + ") AS nSumTotal \
       FROM BalanceSheets \
       GROUP BY \
         Year, \
         Quarter, \
         Country, \
         Company \
       HAVING \
          Year=" + nYear + " \
       AND \
          Quarter=" + nQuarter + " \
       AND \
          Country='" + sCountry + "' \
       AND \
          Company='" + sCompany + "' \
      ;"
   sSQL=re.sub("\s\s+", " ", sSQL)
   sSumRaw=pd.read_sql_query(sSQL, conn)
   ws[sCellField] = sSumRaw["nSumTotal"][0] * nSumSi
   print('Set cell',sCellField,' to ', sSumField,'Total')
   wb.save(sFileName)
```

```
##############################
Loading : C:/VKHCG/04-Clark/00-RawData/BalanceSheets09.csv
##############################
##############################
Storing : C:/VKHCG/04-Clark/06-Report/SQLite/clark.db  Table: BalanceSheets
Append Data
##############################
Loading : C:/VKHCG/04-Clark/00-RawData/BalanceSheets10.csv
##############################
##############################
Storing : C:/VKHCG/04-Clark/06-Report/SQLite/clark.db  Table: BalanceSheets
Append Data
##############################
Loading : C:/VKHCG/04-Clark/00-RawData/BalanceSheets11.csv
##############################
##############################
Storing : C:/VKHCG/04-Clark/06-Report/SQLite/clark.db  Table: BalanceSheets
Append Data
##############################
Loading : C:/VKHCG/04-Clark/00-RawData/BalanceSheets12.csv
##############################
##############################
Storing : C:/VKHCG/04-Clark/06-Report/SQLite/clark.db  Table: BalanceSheets
Append Data
(71136, 7)
c:\Users\krsna\Desktop\DS practical\p9\4.py:73: DeprecationWarning: Call to deprecated function get_sheet_by_name (Use wb[sheetname]).
  ws=wb.get_sheet_by_name("Balance-Sheet")
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Clark-Afghanistan.xlsx
Set cell H30  to  Capital_Stock Total
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Hillman-Afghanistan.xlsx
Set cell H30  to  Capital_Stock Total
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Krennwallner-Afghanistan.xlsx
Set cell H30  to  Capital_Stock Total
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Vermeulen-Afghanistan.xlsx
Set cell H30  to  Capital_Stock Total
C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/Report-Balance-Sheet-2000Q1-Clark-AlandIslands.xlsx
Set cell H30  to  Capital_Stock Total
PS C:\Users\krsna\Desktop\DS practical>
```

Practical 8

Practical 8: Data Visualization with Power BI

Task 1: Import order data from an OData feed

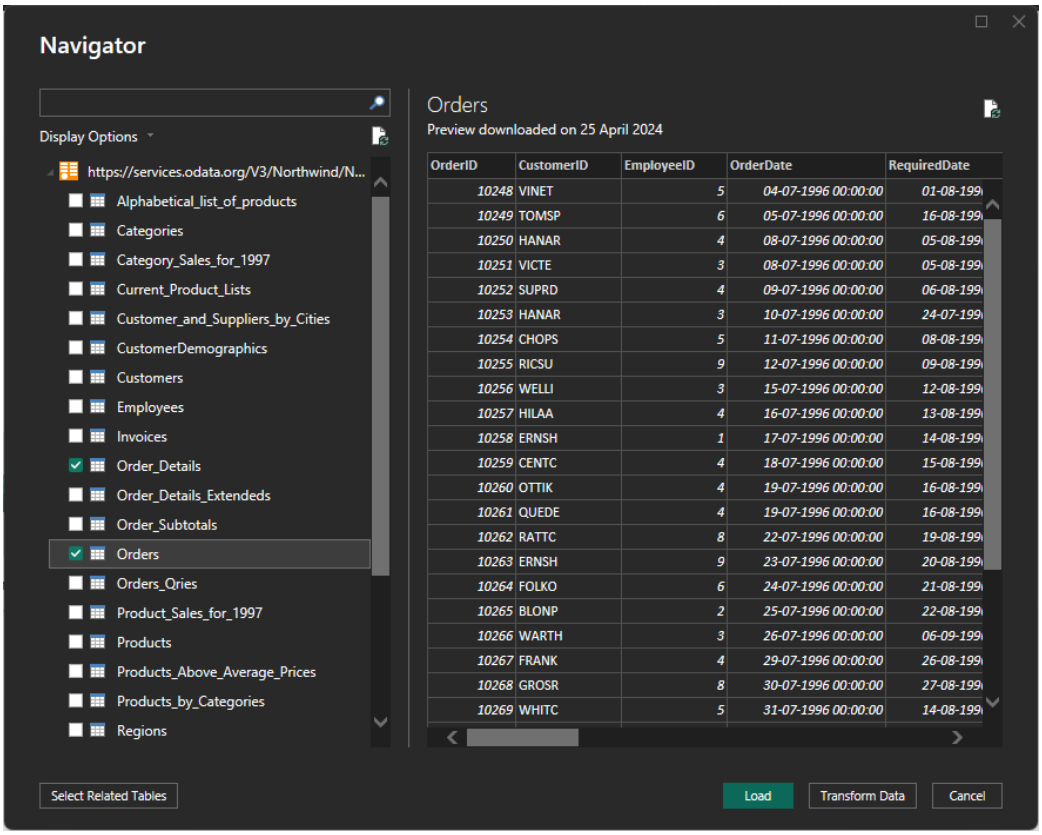
You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below: http://services.odata.org/V3/Northwind/Northwind.svc/

Step 1: Connect to an OData feed

1. From the Home ribbon tab in Query Editor, select Get Data.

2. Browse to the OData Feed data source.

3. In the OData Feed dialog box, paste the URL for the Northwind OData feed. 4. Select OK.



Step 2: Expand the Order_Details table

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

The Expand operation combines columns from a related table into a subject table. When the query runs, rows from the related table (Order_Details) are combined into rows from the subject table (Orders).

After you expand the Order_Details table, three new columns and additional rows are added to the Orders table, one for each row in the nested or related table.

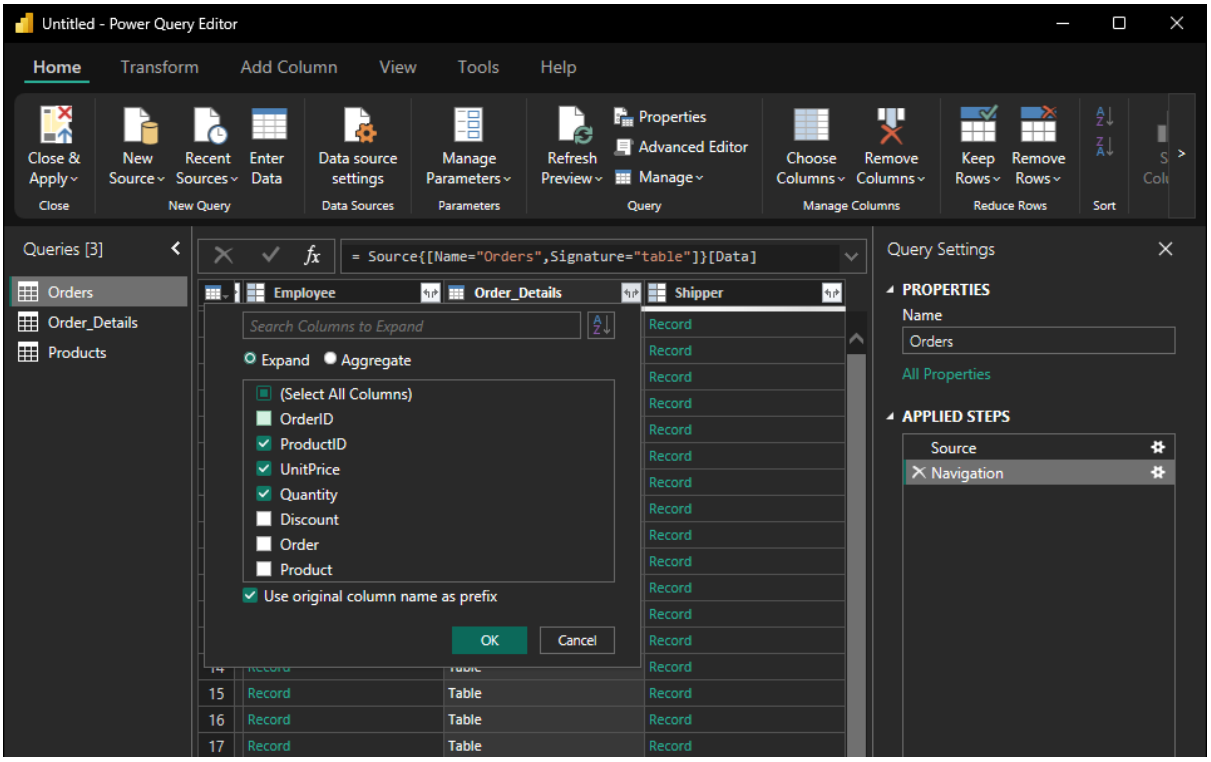
1. In the Query View, scroll to the Order_Details column.

2. In the Order_Details column, select the expand icon ().

3. In the Expand drop-down: a. Select (Select All Columns) to clear all columns.

Select ProductID, UnitPrice, and Quantity.

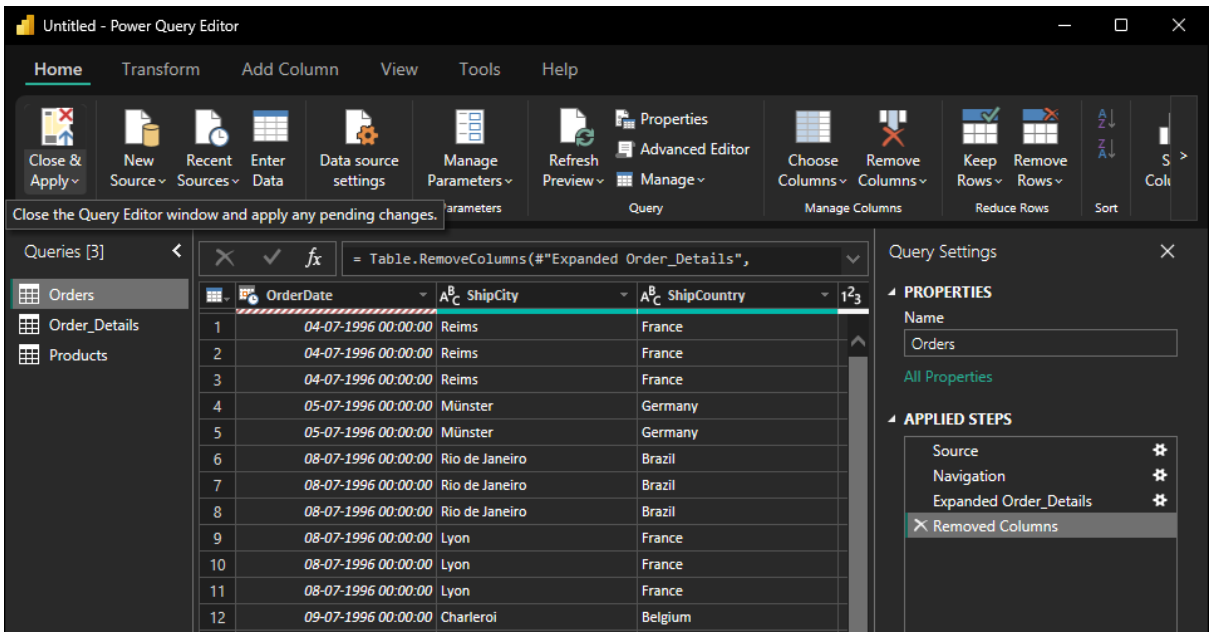
Click OK.



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

In this step you remove all columns except OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and Order_Details.Quantity columns.

In the previous task, you used Remove Other Columns. For this task, you remove selected columns



Step 4: Calculate the line total for each Order_Details row

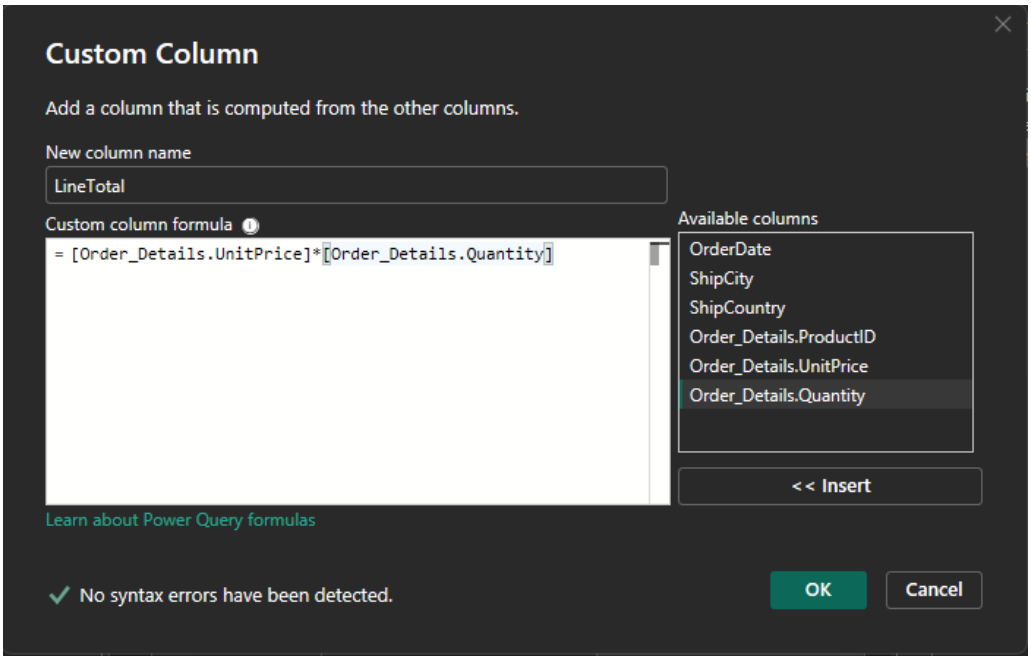
Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order_Details row.

Calculate the line total for each Order_Details row:

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

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

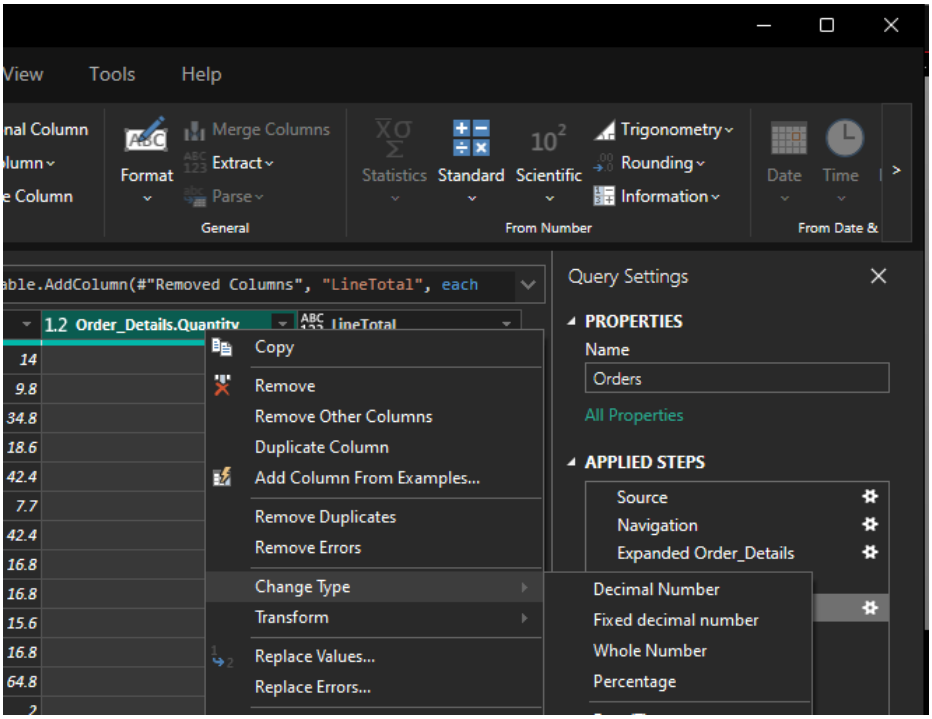
3. In the New column name textbox, enter LineTotal.

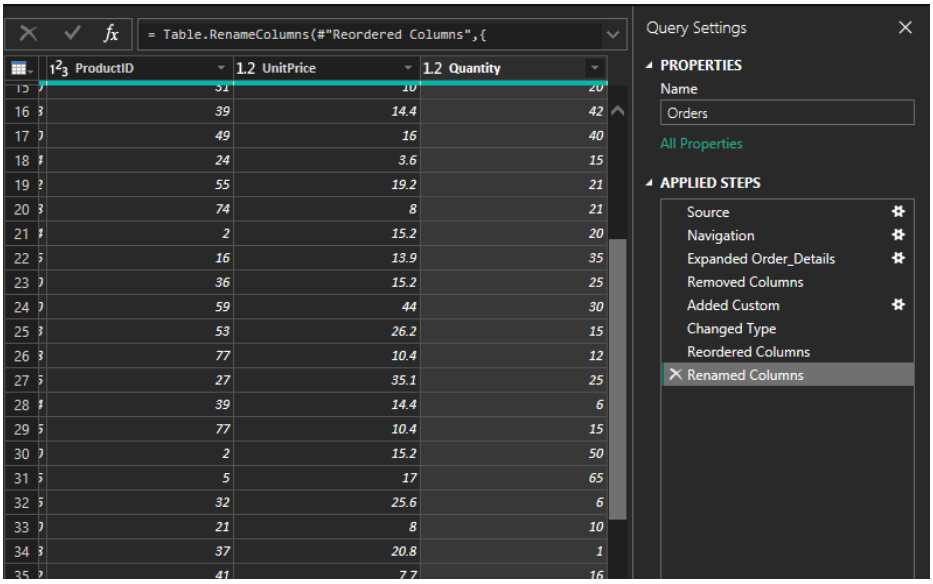


Step 5: Set the datatype of the LineTotal field

1. Right click the LineTotal column.

2. Select Change Type and choose Decimal Number.





Step 6: Rename and reorder columns in the query

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

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

Task 2: Combine the Products and Total Sales queries

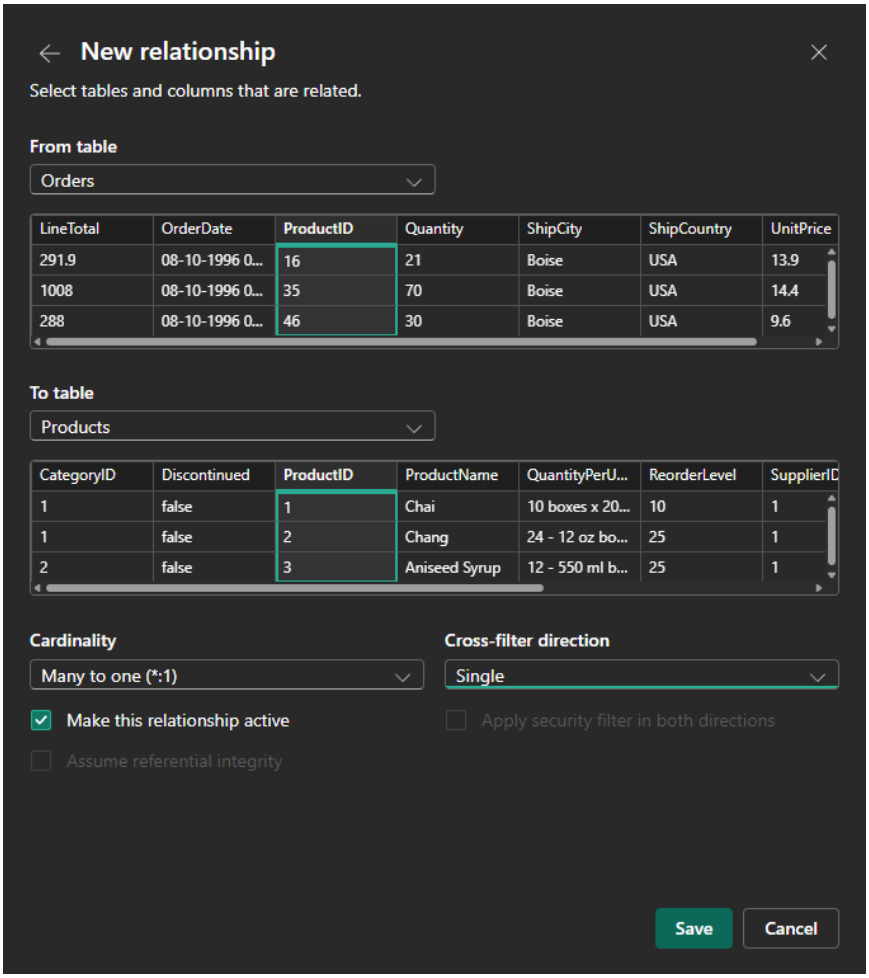
1. Confirm the relationship between Products and Total sales

2. Power BI Desktop loads the data from the two queries

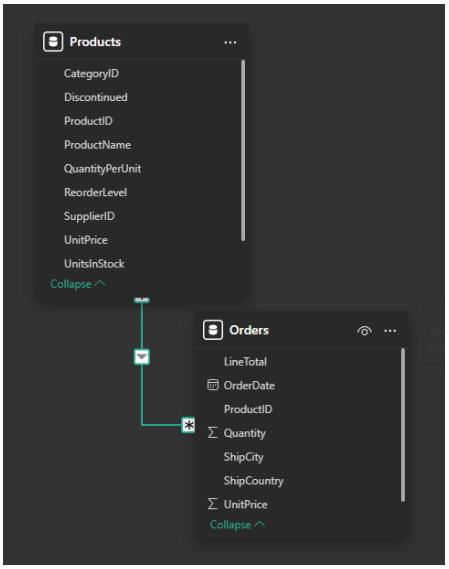
3. Once the data is loaded, select the Manage Relationships button Home ribbon

4. Select the New… button

5. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.

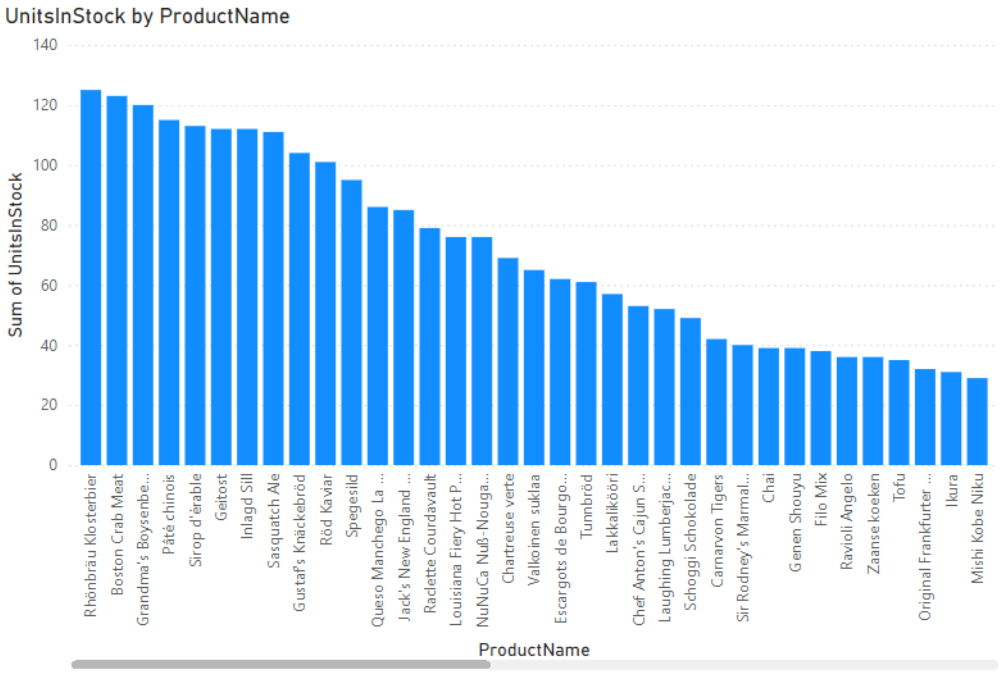


6. Select Cancel, and then select Relationship view in Power BI Desktop

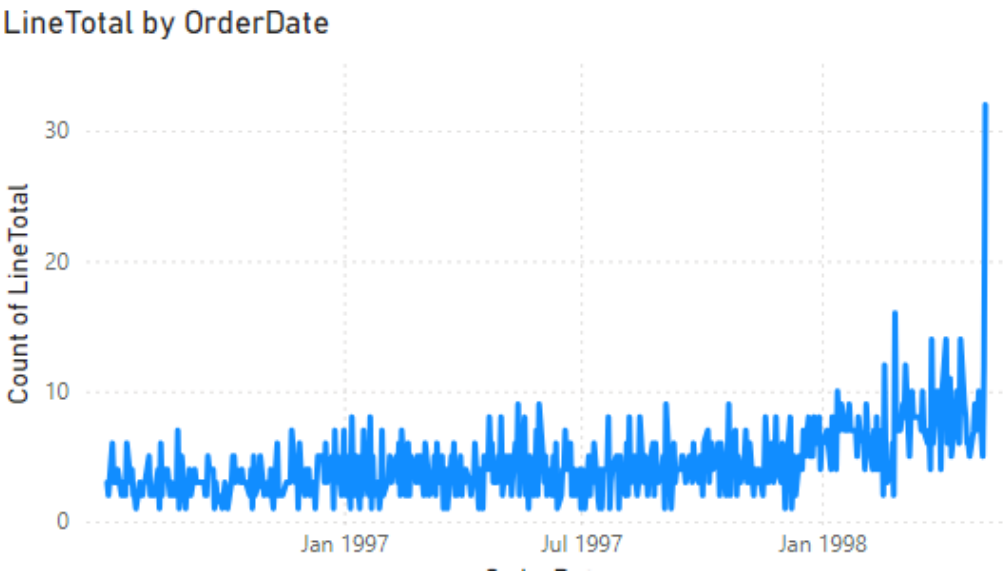


Task 3: Build visuals using your data

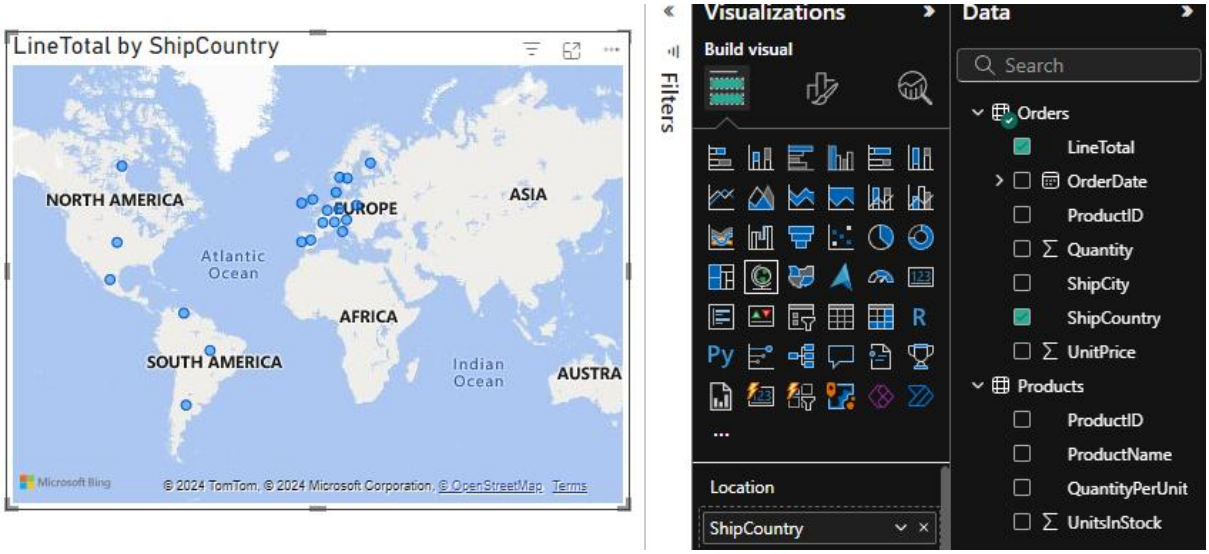
Step 1: Create charts showing Units in Stock by Product and Total Sales by Year



Step 2: Drag OrderDate to the canvas beneath the first chart , then drag LineTotal on to the visual, then Select Line chart. The following visaualization is created.



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



Step 4. Interact with your report visuals to analyse further