Practical 1A: Text delimited CSV to HORUS format.

Code:

```
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')
```

Output:

```
--------------- RESTART: C:\VKHCG\05-DS\9999-Data\CSV2HORUS.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]
=====================================================
CSV to HORUS - Done
>>>
```

Practical 1B: XML to HORUS Format

Code:

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

Output:

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

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

Practical 1C: JSON to HORUS Format.

Code:

```
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')
```

Output:

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

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

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

Practical 1D: MySql Database to HORUS Format

Code:

```
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')
```

Output:

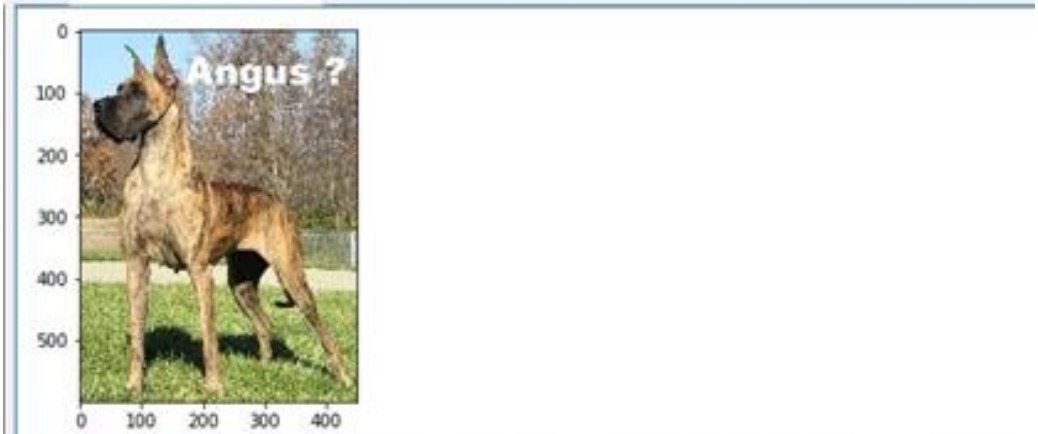
```
>>>
========== RESTART: C:\VKHCG\05-DS\9999-Data\DATABASE2HORUS.py ==========
Input Data Values ===================================
     index                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0        0                Afghanistan          AF         AFG        4
1        1             Aland 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
>>>
```

Practical 1E: Picture (JPEG) to HORUS Format

Code:

```
from scipy.misc import imread import pandas as pd
import matplotlib.pyplot as plt import numpy as np
# Input Agreement ============================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/Angus.jpg' InputData = imread(sInputFileName,
flatten=False, mode='RGBA')
print('Input Data Values ===================================')
print('X: ',InputData.shape[0]) print('Y: ',InputData.shape[1]) print('RGBA: ', InputData.shape[2])
print('=====================================================')  # Processing
Rules ============================================
ProcessRawData=InputData.flatten() y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y))) sColumns=
['XAxis','YAxis','Red', 'Green', 'Blue','Alpha'] ProcessData.columns=sColumns
ProcessData.index.names =['ID'] print('Rows: ',ProcessData.shape[0]) print('Columns
:',ProcessData.shape[1])
print('=====================================================')
print('Process Data Values ================================')
print('=====================================================')
plt.imshow(InputData) plt.show()
print('=====================================================')  # Output
Agreement ============================================
OutputData=ProcessData print('Storing File')
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Picture to HORUS - Done')
print('=====================================================')
```

Output:




```
==========================================================
Storing File
==========================================================
Picture to HORUS - Done
==========================================================
```

Practical 1F: ideo to HORUS Format

Code:

(Part 1)

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

Output:

```
>>>
========== RESTART: C:\VKHCG\05-DS\9999-Data\MOVIE2HORUSFrame.py ==========
================================================
Start Movie to Frames
================================================
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0000.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0001.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0002.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0003.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0004.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0005.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0006.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0007.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0008.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0009.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0010.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0099.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0100.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0101.jpg
================================================
Generated :  101  Frames
================================================
Movie to Frames HORUS - Done
================================================
>>>
```

(Part 2)

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

```
print('Movie to HORUS - Done')
print('=======================================================')
```

Output:

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

dog-frame-0000.jpeg

dog-frame-0001.jpeg

dog-frame-0100.jpeg

dog-frame-0101.jpeg

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

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

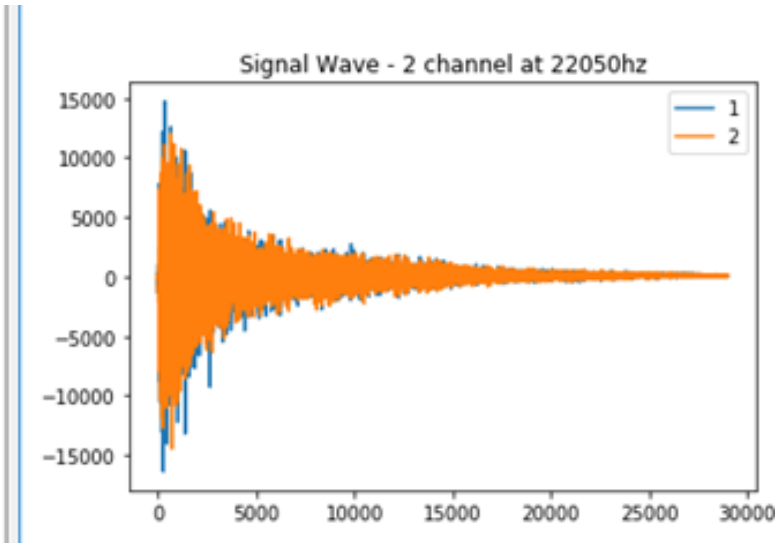
Practical 1G: Audio to HORUS Format

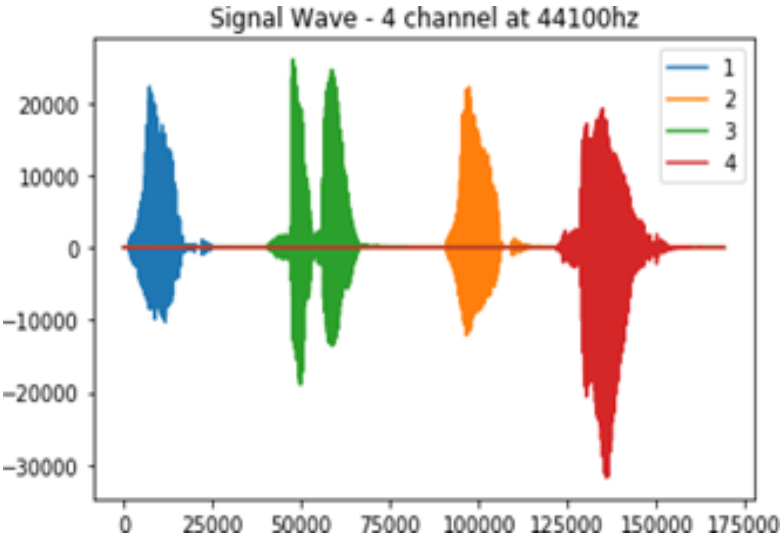
Code:

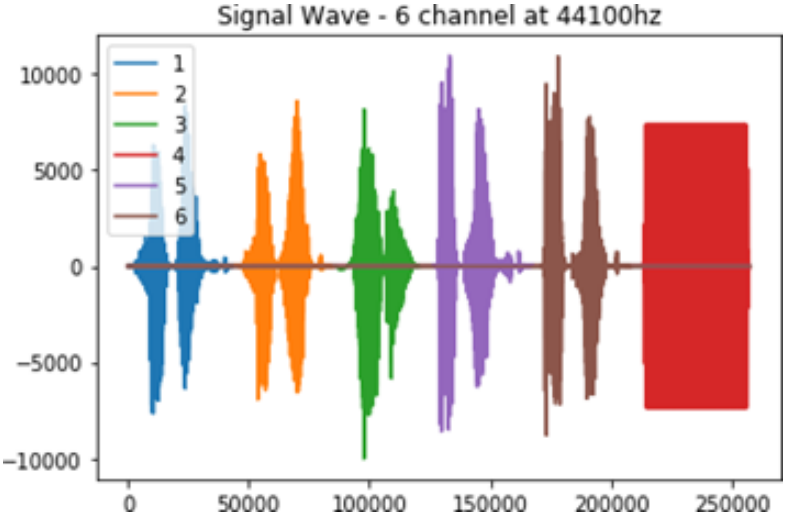
```
from scipy.io import wavfile import pandas as pd
import matplotlib.pyplot as plt import numpy as np
#============================================================
def show_info(aname, a,r): print ('      ')
print ("Audio:", aname) print ('           ')
print ("Rate:", r) print ('          ')
print ("shape:", a.shape)
print ("dtype:", a.dtype)
print ("min, max:", a.min(), a.max()) print ('   ')
plot_info(aname, a,r)
#============================================================
def plot_info(aname, a,r):
sTitle= 'Signal Wave - '+ aname + ' at ' + str(r) + 'hz' plt.title(sTitle)
sLegend=[]
for c in range(a.shape[1]): sLabel = 'Ch' + str(c+1) sLegend=sLegend+[str(c+1)] plt.plot(a[:,c],
label=sLabel)
plt.legend(sLegend) plt.show()
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'
print('============================================================')
print('Processing : ', sInputFileName)
print('============================================================')
InputRate, InputData = wavfile.read(sInputFileName) show_info("2 channel",
InputData,InputRate) ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2'] ProcessData.columns=sColumns OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('============================================================')
print('Processing : ', sInputFileName)
print('============================================================')
InputRate, InputData = wavfile.read(sInputFileName) show_info("4 channel",
InputData,InputRate) ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4'] ProcessData.columns=sColumns OutputData=ProcessData
```

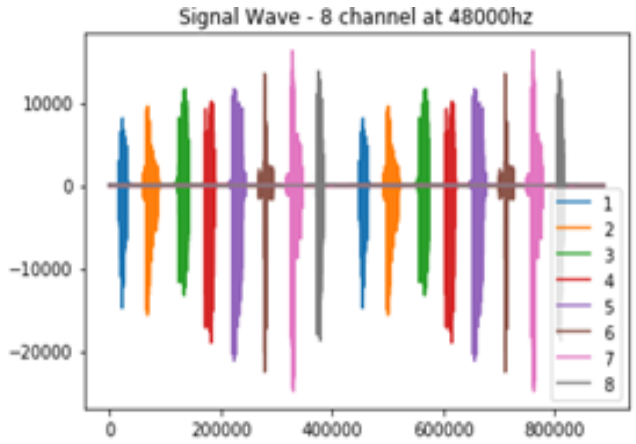
```
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#================================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('=============================================================')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

Output:


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


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


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



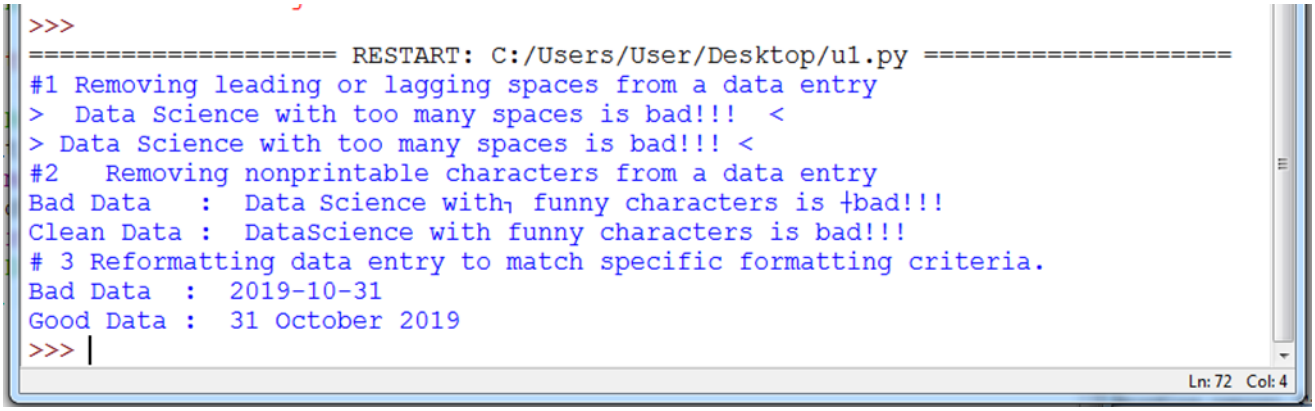

```
==========================================================
Audio to HORUS - Done
==========================================================
```

Practical 2A: Fixers Utilities

Code:

```
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print('#1 Removing leading or lagging spaces from a data entry'); baddata = " Data Science with too many spaces is bad!!! " print('>',baddata,'<')
cleandata=baddata.strip() print('>',cleandata,'<')

# 2  Removing nonprintable characters from a data entry
print('#2  Removing nonprintable characters from a data entry') printable = set(string.printable)
baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!" cleandata='
'.join(filter(lambda x: x in string.printable,baddata)) print('Bad Data  : ',baddata);
print('Clean Data : ',cleandata)

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

Output:

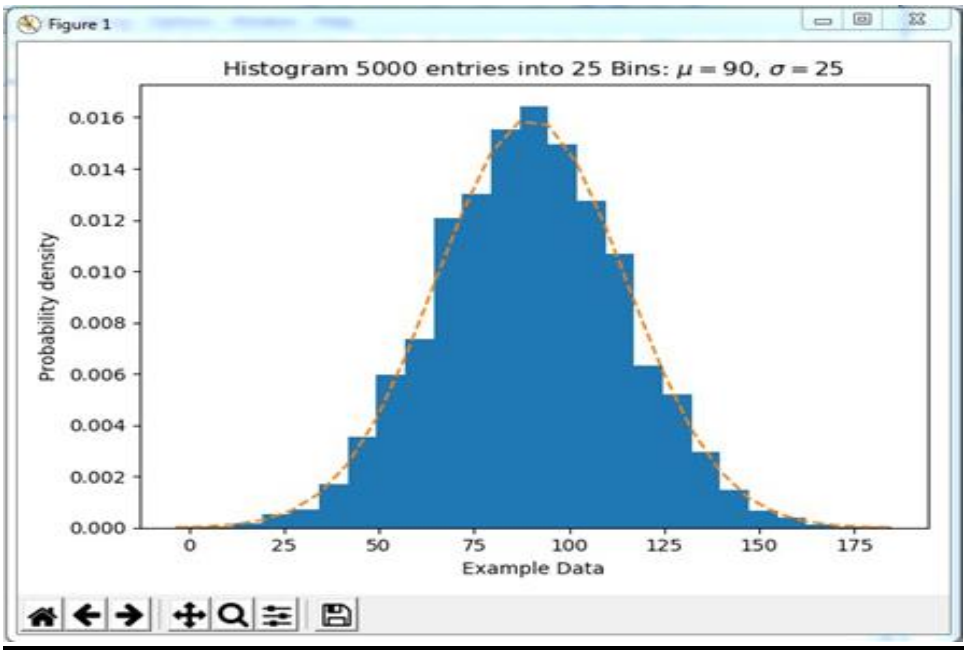


Practical 2B: Data Binning or Bucketing

Code:

```
import numpy as np
import matplotlib.mlab as mlab import matplotlib.pyplot as plt import scipy.stats as stats
np.random.seed(0) # 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()
```

Output:



Practical 2C: Averaging of Data

Code:

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

Output:

```
>>>
========== RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Mean.py ==========
#################################
Working Base : C:/VKHCG  using
#################################
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
```

Practical 2D: utlier Detection

Code:

```
import pandas as pd
#########################################################################
InputFileName='IP_DATA_CORE.csv' OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG' print('##################################')
print('Working Base :',Base) print('##################################')
#########################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data') print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std() print('Outliers')
UpperBound=float(MeanData+StdData) print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound] print(OutliersHigher)
LowerBound=float(MeanData-StdData) print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound] print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) & (AllData.Latitude<=UpperBound)]
print(OutliersNot)
```

Output:

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

```
1911    GB    London   51.5508
1912    GB    London   51.5649
1913    GB    London   51.5895
1914    GB    London   51.5232
...     ...       ...      ...
3434    GB    London   51.5092
3435    GB    London   51.5092
3436    GB    London   51.5163
3437    GB    London   51.5085
3438    GB    London   51.5136

[1502 rows x 3 columns]
Outliers
Higher than 51.51263550786781
     Country Place_Name  Latitude
1910    GB    London   51.5130
1911    GB    London   51.5508
1912    GB    London   51.5649
1913    GB    London   51.5895
1914    GB    London   51.5232
1916    GB    London   51.5491
1919    GB    London   51.5161
1920    GB    London   51.5198
1921    GB    London   51.5198
1923    GB    London   51.5237
1924    GB    London   51.5237
1925    GB    London   51.5237
1926    GB    London   51.5237
1927    GB    London   51.5232
3436    GB    London   51.5163
3438    GB    London   51.5136
Lower than 51.50617687562166
     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]
```

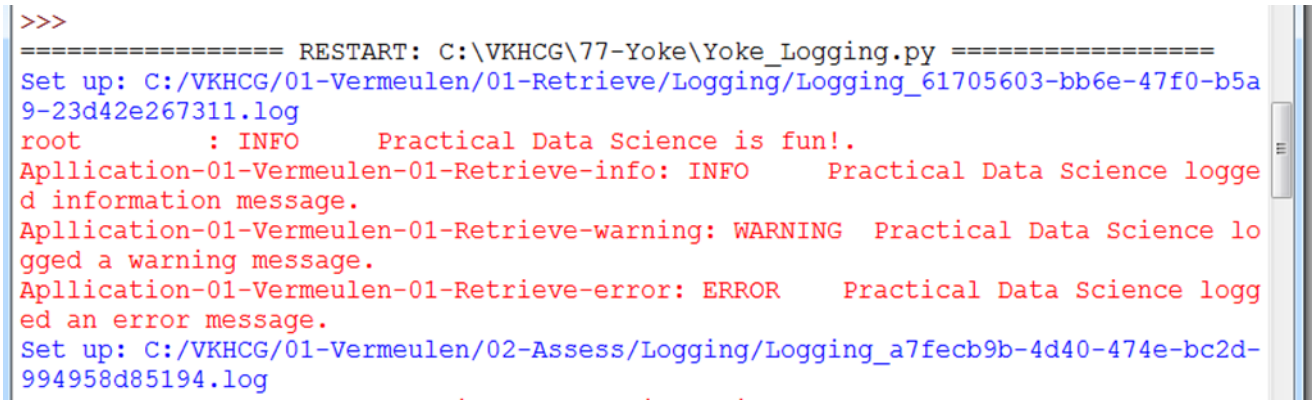
Practical 2E: Logging

Code:

```
import sys import os import logging import uuid import shutil import time
###################################################################### Base='C:/VKHCG'
######################################################################
sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04-Clark']
sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-Organise','06-Report']
sLevels=['debug','info','warning','error']

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

for sLevel in sLevels:
sApp='Apllication-'+ sCompany + '-' + sLayer + '-' + sLevel logger = logging.getLogger(sApp)
if sLevel == 'debug':
```

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

Output:



Practical 3A: Perform the following data processing using R.

Code:

Use R-Studio for the following:

```
>library(readr)
  Warning message: package 'readr' was built under R version 3.4.4
```

Load a table named IP_DATA_ALL.csv.

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

IP_DATA_COUNTRY_FREQ

Filter

	Country	N
2	GB	1502
3	US	1383
1	DE	677

```
hist_latitude =data.table(Latitude=unique(IP_DATA_ALL_FIX [is.na(IP_DATA_ALL_with_ID
['Latitude']) == 0, ]$Lati tude))
setkeyv(hist_latitude, 'Latitude') setorder(hist_latitude)
hist_latitude_with_id=rowid_to_column(hist_latitude, var = "RowID")
View(hist_latitude_with_id)
IP_DATA_Latitude_FREQ=data.table(with(IP_DATA_ALL_FIX,table(Latitude)))
View(IP_DATA_Latitude_FREQ)
```

	Latitude	N
133	51.5092	1478
1	40.6888	130
107	48.15	114
9	40.7143	113

```
>sapply(IP_DATA_ALL_FIX[,'Latitude'], min, na.rm=TRUE)
Latitude 40.6888
>sapply(IP_DATA_ALL_FIX[,'Country'], min, na.rm=TRUE)
Country "DE"
```

Minimum business frequency is from DE – Denmark.

```
>sapply(IP_DATA_ALL_FIX[,'Latitude'], max, na.rm=TRUE)
Latitude
51.5895
>sapply(IP_DATA_ALL_FIX[,'Country'], max, na.rm=TRUE)
Country "US"
>sapply(IP_DATA_ALL_FIX [,'Latitude'], mean, na.rm=TRUE)
Latitude
46.69097
>sapply(IP_DATA_ALL_FIX [,'Latitude'], median, na.rm=TRUE)
Latitude
48.15
```

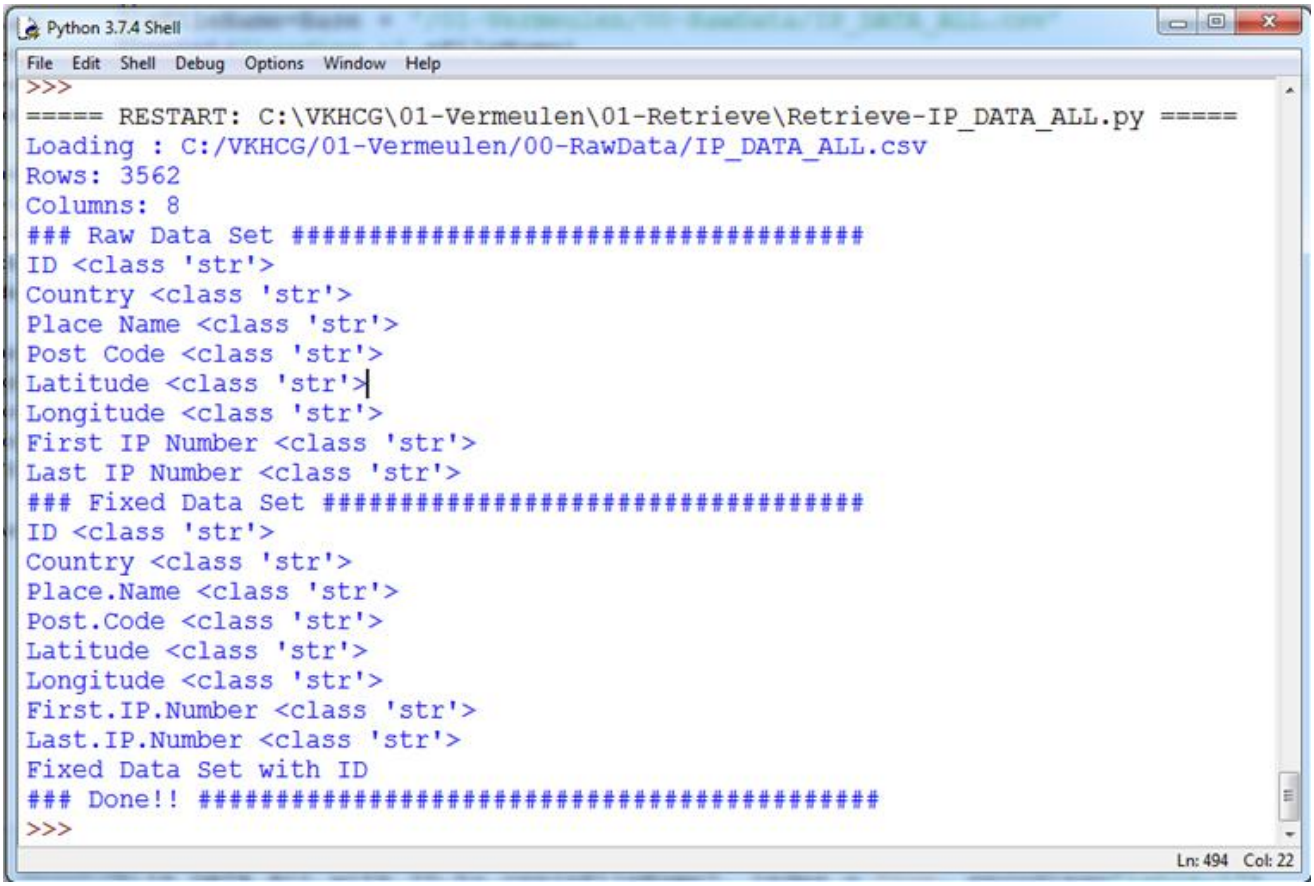
```
>sapply(IP_DATA_ALL_FIX [,'Latitude'], range, na.rm=TRUE)
Latitude
[1,] 40.6888
[2,] 51.5895
>sapply(IP_DATA_ALL_FIX [,'Latitude'], quantile, na.rm=TRUE)
Latitude
0%      40.6888
25% 40.7588
50% 48.1500
75% 51.5092
100% 51.5895
>sapply(IP_DATA_ALL_FIX [,'Latitude'], sd, na.rm=TRUE)
Latitude
4.890387
>sapply(IP_DATA_ALL_FIX [,'Longitude'], sd, na.rm=TRUE)
Longitude
38.01702
```

Practical 3B: rogram to retrieve different attributes of data.

Code:

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

Output:


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-IP_DATA_ALL.py =====
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 3562
Columns: 8
Raw Data Set
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
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!!
>>>
Ln: 494 Col: 22

Practical 3C: Data Pattern

Code:

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

Output:

Filter

	Country	PatternCountry
1	US	AA
2	DE	AA
3	GB	AA

Practical 3D: Loading IP_DATA_ALL

Code:

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

Output:

```
>>>
===== RESTART: C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-IP_DATA_ALL.py =====
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 3562
Columns: 8
### Raw Data Set #####################################
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 ###################################
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!! ############################################
>>>
```

Practical 4A: Perform error management on the given data using pandas package.

I Drop the Columns Where All Elements Are Missing Values

Code:

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

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

Output:

```
======= RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-Good-Bad-01.py =======
###############################
Working Base :C:/VKHCG using win32
###############################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
###############################
## Raw Data Values
###############################
    ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0   1.0   Good  Better   Best  1024.0    NaN   10241.0       1
1   2.0   Good     NaN   Best   512.0    NaN    5121.0       2
2   3.0   Good  Better    NaN  256.0     NaN    256.0       3
3   4.0   Good  Better   Best    NaN     NaN    211.0       4
4   5.0   Good  Better    NaN   64.0     NaN   6411.0       5
5   6.0   Good     NaN   Best   32.0     NaN     32.0       6
6   7.0    NaN  Better   Best   16.0     NaN   1611.0       7
7   8.0    NaN     NaN   Best    8.0     NaN   8111.0       8
8   9.0    NaN     NaN    NaN    4.0     NaN     41.0       9
9  10.0      A       B      C    2.0     NaN  21111.0      10
10  NaN    NaN     NaN    NaN    NaN     NaN      NaN      11
11 10.0   Good  Better   Best  1024.0    NaN  102411.0      12
12 10.0   Good     NaN   Best   512.0    NaN     512.0     13
13 10.0   Good  Better    NaN  256.0     NaN    1256.0     14
14 10.0   Good  Better   Best    NaN     NaN      NaN      15
15 10.0   Good  Better    NaN   64.0     NaN    164.0      16
16 10.0   Good     NaN   Best   32.0     NaN    322.0      17
17 10.0    NaN  Better   Best   16.0     NaN    163.0      18
18 10.0    NaN     NaN   Best    8.0     NaN    844.0      19
19 10.0    NaN     NaN    NaN    4.0     NaN   4555.0      20
20 10.0      A       B      C    2.0     NaN    111.0      21
```

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

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

Code:

```
import sys import os
import pandas as pd
#####################################################################
Base='C:/VKHCG'
sInputFileName='Good-or-Bad.csv' sOutputFileName='Good-or-Bad-02.csv' Company='01-
Vermeulen'
#####################################################################
Base='C:/VKHCG'
#####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
### Import Warehouse
#####################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName print('Loading
:',sFileName) RawData=pd.read_csv(sFileName,header=0)

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

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

Output:

```
======= RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-Good-Bad-02.py =======
################################
Working Base : C:/VKHCG using win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
################################
## Raw Data Values
################################
    ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0   1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1   2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2   3.0   Good  Better    NaN   256.0     NaN     256.0       3
3   4.0   Good  Better   Best     NaN     NaN     211.0       4
4   5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5   6.0   Good     NaN   Best    32.0     NaN      32.0       6
6   7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7   8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8   9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9  10.0      A       B      C     2.0     NaN   21111.0      10
10  NaN    NaN     NaN    NaN    NaN     NaN       NaN      11
11 10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12 10.0   Good     NaN   Best   512.0     NaN     512.0      13
13 10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14 10.0   Good  Better   Best     NaN     NaN       NaN      15
15 10.0   Good  Better    NaN    64.0     NaN     164.0      16
16 10.0   Good     NaN   Best    32.0     NaN     322.0      17
17 10.0    NaN  Better   Best    16.0     NaN     163.0      18
```

```
 18  10.0   NaN   NaN  Best    8.0   NaN   844.0    19
 19  10.0   NaN   NaN   NaN    4.0   NaN  4555.0    20
 20  10.0     A     B     C    2.0   NaN   111.0    21

## Data Profile
#################################
Rows : 21
Columns : 8
#################################
#################################
## Test Data Values
#################################
   FieldG
0       1
1       2
2       3
3       4
4       5
5       6
6       7
7       8
8       9
9      10
10     11
11     12
12     13
13     14
14     15
15     16
16     17
17     18
18     19
19     20
20     21
#################################
## Data Profile
#################################
Rows : 21
Columns : 1
#################################
#################################
### Done!! ####################
#################################
>>>
```

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

Code:

```
import sys import os
import pandas as pd
#####################################################################
sInputFileName='Good-or-Bad.csv' sOutputFileName='Good-or-Bad-03.csv' Company='01-
Vermeulen' Base='C:/VKHCG'
#####################################################################
print('##################################')
print('Working Base :',Base, ' using Windows ~~~~')
print('##################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
### Import Warehouse
#####################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName print('Loading
:',sFileName) RawData=pd.read_csv(sFileName,header=0)

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

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

Output:

Before

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

After

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

Practical 4B: Write Python / R program to create the network routing diagram from the given data on routers.

Code:

```
import sys
import os
import pandas as pd
#######################################################################
pd.options.mode.chained_assignment = None
#######################################################################
Base='C:/VKHCG'
#######################################################################
print('###################################')
print('Working Base :',Base, ' using Windows') print('###################################')
#######################################################################
sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sInputFileName3='01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv'
#######################################################################
sOutputFileName='Assess-Network-Routing-Company.csv' Company='01-Vermeulen'
#######################################################################
#######################################################################
### Import Country Data
#######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('###################################')
print('Loading :',sFileName) print('###################################')
CountryData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
print('###################################')
#######################################################################
## Assess Country Data
#######################################################################
print('###################################')
print('Changed :',CountryData.columns.values) CountryData.rename(columns={'Country':
'Country_Name'}, inplace=True) CountryData.rename(columns={'ISO-2-CODE':
'Country_Code'}, inplace=True) CountryData.drop('ISO-M49', axis=1, inplace=True)
```

```
CountryData.drop('ISO-3-Code', axis=1, inplace=True) CountryData.drop('RowID', axis=1,
inplace=True) print('To :',CountryData.columns.values)
print('#################################')
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('#################################')
print('Loading :',sFileName) print('#################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('#################################')
#########################################################################
## Assess Company Data
#########################################################################
print('#################################')
print('Changed :',CompanyData.columns.values) CompanyData.rename(columns={'Country':
'Country_Code'}, inplace=True) print('To :',CompanyData.columns.values)
print('#################################')
#########################################################################
#########################################################################
### Import Customer Data
#########################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName3
print('#################################')
print('Loading :',sFileName) print('#################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('#################################')
print('Loaded Customer :',CustomerRawData.columns.values)
print('#################################')
#########################################################################
CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('#################################')
print('Remove Blank Country Code')
print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])
print('#################################')
#########################################################################
print('#################################')
print('Changed :',CustomerData.columns.values) CustomerData.rename(columns={'Country':
'Country_Code'}, inplace=True) print('To :',CustomerData.columns.values)
print('#################################')
#########################################################################
print('#################################')
```

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

Output:

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

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

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

Output

Assess-Network-Routing-Customer.csv

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

Assess-Network-Routing-Node.py

```
#######################################################################
import sys import os
import pandas as pd
#######################################################################
pd.options.mode.chained_assignment = None
#######################################################################
Base='C:/VKHCG'
#######################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#######################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
#######################################################################
sOutputFileName='Assess-Network-Routing-Node.csv' Company='01-Vermeulen'
#######################################################################
### Import IP Data
#######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName) print('#################################')
IPData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded IP :', IPData.columns.values) print('#################################')
#######################################################################
print('#################################')
print('Changed :',IPData.columns.values) IPData.drop('RowID', axis=1, inplace=True)
IPData.drop('ID', axis=1, inplace=True)
IPData.rename(columns={'Country': 'Country_Code'}, inplace=True)
IPData.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
IPData.rename(columns={'Post.Code': 'Post_Code'}, inplace=True)
IPData.rename(columns={'First.IP.Number': 'First_IP_Number'}, inplace=True)
IPData.rename(columns={'Last.IP.Number': 'Last_IP_Number'}, inplace=True) print('To
:',IPData.columns.values) print('#################################')
#######################################################################
print('#################################')
print('Change ',IPData.columns.values) for i in IPData.columns.values:
j='Node_'+i
```

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

Output:

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

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

Practical No 4C: Write a Python / R program to build directed acyclic graph.

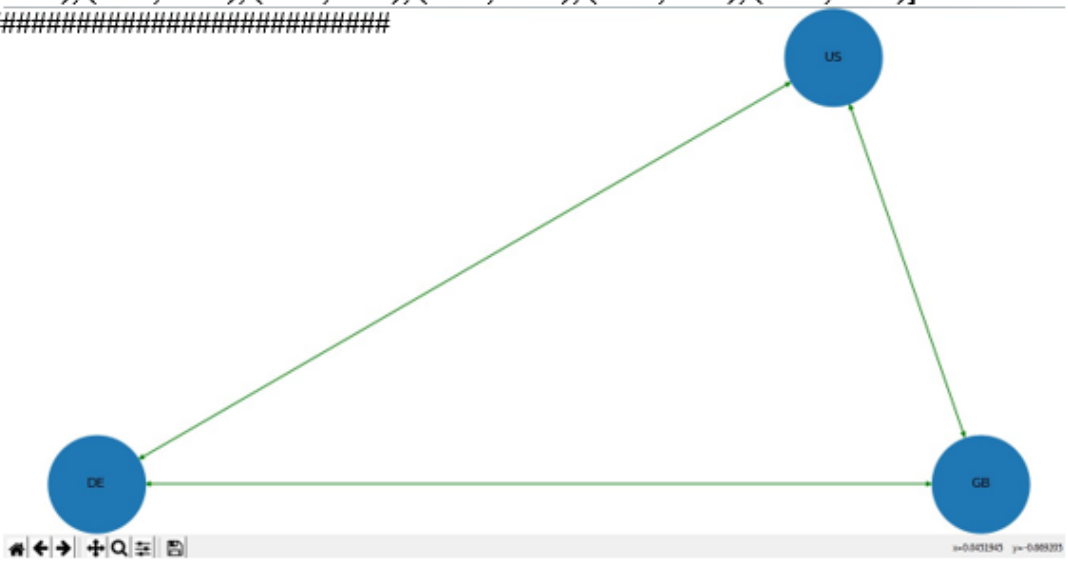
Code:

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

```
print('Link :',n1,' to ', n2) G1.add_edge(n1,n2)
print('##################################')

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

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

Output:

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



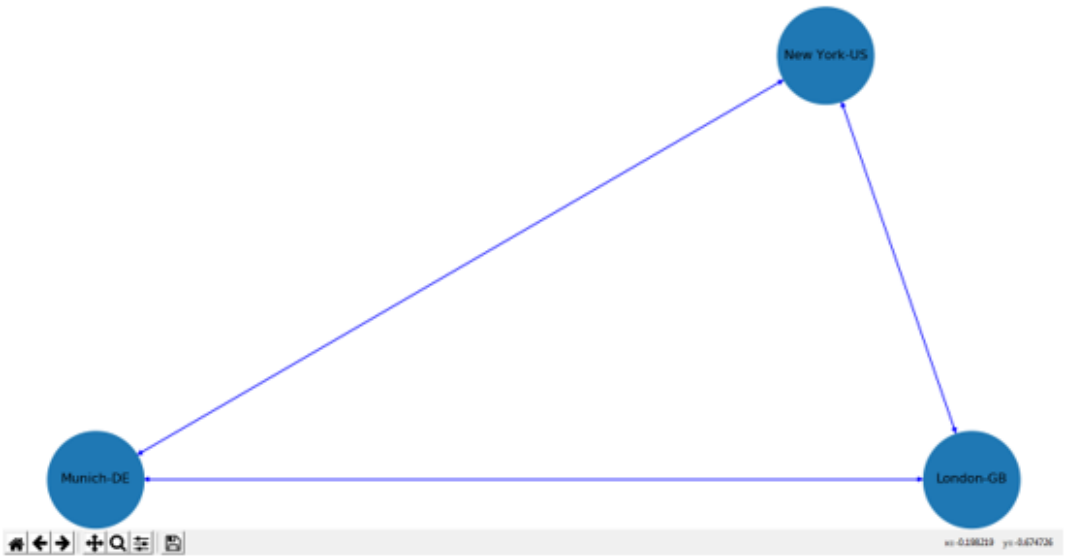
Customer Location DAG

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

print('##################################')
for n1 in G1.nodes(): for n2 in G1.nodes():
if n1 != n2:
print('Link :',n1,' to ', n2) G1.add_edge(n1,n2)
```

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

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

Output:

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



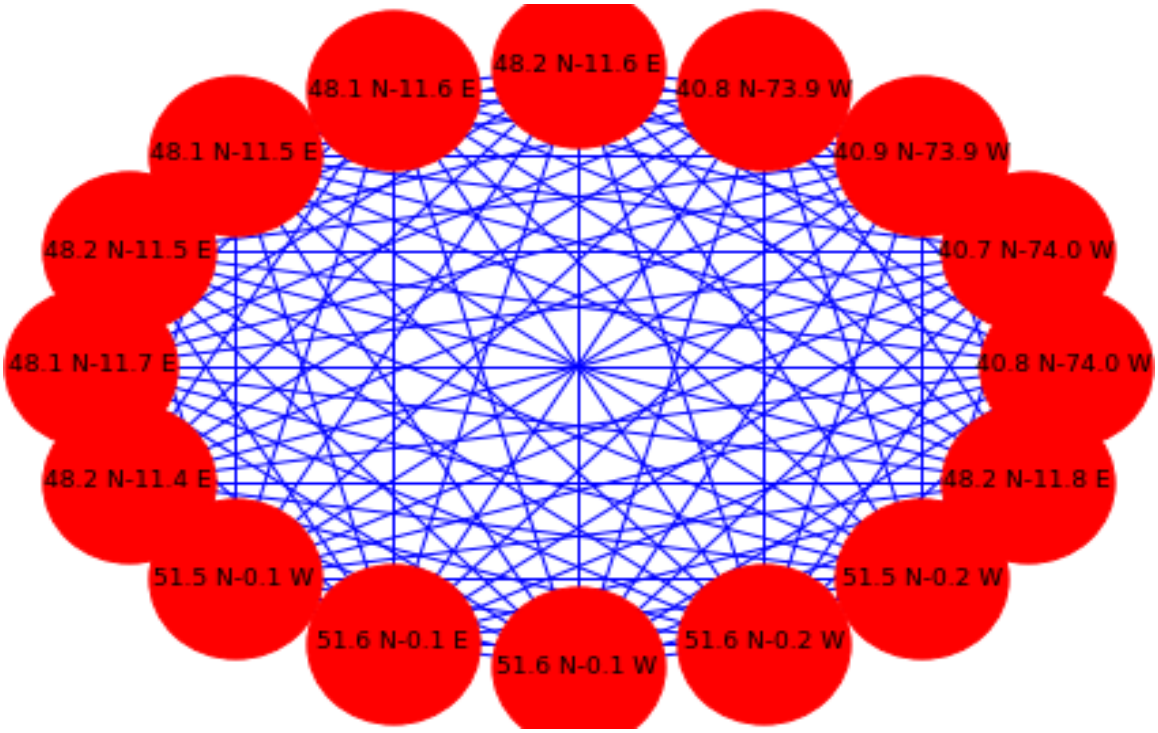
Open your Python editor and create a file named Assess-DAG-GPS.py in directory C:\VKHCG\01-Vermeulen\02-Assess.

```
import networkx as nx
import matplotlib.pyplot as plt import sys
import os
import pandas as pd
ase='C:/VKHCG' print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName='Assess-DAG-Company-GPS.png'
Company='01-Vermeulen' ### Import Company Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName) print('##################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('##################################')
print(CompanyData) print('##################################')
print('Rows : ',CompanyData.shape[0]) print('##################################')
G=nx.Graph()
for i in range(CompanyData.shape[0]): nLatitude=round(CompanyData['Latitude'][i],2)
nLongitude=round(CompanyData['Longitude'][i],2)

if nLatitude < 0:
sLatitude = str(nLatitude*-1) + ' S' else:
sLatitude = str(nLatitude) + ' N'

if nLongitude < 0:
sLongitude = str(nLongitude*-1) + ' W' else:
sLongitude = str(nLongitude) + ' E'

sGPS= sLatitude + '-' + sLongitude G.add_node(sGPS)

print('##################################')
for n1 in G.nodes(): for n2 in G.nodes():
if n1 != n2:
print('Link :',n1,' to ', n2) G.add_edge(n1,n2)
print('##################################')
```

```
print('###################################')
print("Nodes of graph: ") print(G.number_of_nodes())
print("Edges of graph: ") print(G.number_of_edges())
print('###################################')
```

Output:

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



Practical 4D: Write a Python / R program to pick the content for Bill Boards from the given data.

Code:

Picking Content for Billboards
The basic process required is to combine two sets of data and then calculate the number of visitors per day from the range of IP addresses that access the billboards in Germany.
Bill Board Location: Rows - 8873 Access Visitors: Rows - 75999
Access Location Record: Rows – 1, 81, 235
Open Python editor and create a file named **Assess-DE-Billboard.py** in directory C:\VKHCG\02-Krennwallner\02-Assess

```
################## Assess-DE-Billboard.py ######################
import sys import os
import sqlite3 as sq import pandas as pd
#######################################################################
Base='C:/VKHCG'
print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
sOutputFileName='Assess-DE-Billboard-Visitor.csv'
Company='02-Krennwallner'
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/krennwallner.db' conn = sq.connect(sDatabaseName)
### Import Billboard Data
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('###################################')
print('Loading :',sFileName) print('###################################')
BillboardRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
BillboardRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BillboardData=BillboardRawData
print('Loaded Company :',BillboardData.columns.values)
print('###################################') print('#################')
sTable='Assess_BillboardData'
print('Storing :',sDatabaseName,' Table:',sTable)
```

Output:

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

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

Retrieve_Online_Visitor

SQLite Visitor's Database

C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db Table:

	BillboardCountry	BillboardPlaceName	...	VisitorLongitude	VisitorYearRate
0	DE	Lake	...	8.5667	26823960.0
1	DE	Horb	...	8.6833	26823960.0
2	DE	Horb	...	8.6833	53753112.0
3	DE	Horb	...	8.6833	107611416.0
4	DE	Horb	...	8.6833	13359384.0

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

Assess-DE-Billboard-Visitor

Practical No 4E: Write a Python / R program to generate GML file from the given csv file.

Code:

```
import networkx as nx
mport sys import os
import sqlite3 as sq import pandas as pd
from geopy.distance import vincenty
#####################################################################
Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#####################################################################
Company='02-Krennwallner' sTable='Assess_BillboardVisitorData' sOutputFileName='Assess-DE-Billboard-Visitor.gml'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db' conn = sq.connect(sDatabaseName)
#####################################################################
print('################')
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="select "
sSQL=sSQL+ " A.BillboardCountry," sSQL=sSQL+ " A.BillboardPlaceName,"
sSQL=sSQL+ " ROUND(A.BillboardLatitude,3) AS BillboardLatitude, " sSQL=sSQL+ "
ROUND(A.BillboardLongitude,3) AS BillboardLongitude,"

sSQL=sSQL+ " (CASE WHEN A.BillboardLatitude < 0 THEN " sSQL=sSQL+ "  'S' ||
ROUND(ABS(A.BillboardLatitude),3)" sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'N' || ROUND(ABS(A.BillboardLatitude),3)" sSQL=sSQL+ " END ) AS
sBillboardLatitude,"

sSQL=sSQL+ " (CASE WHEN A.BillboardLongitude < 0 THEN " sSQL=sSQL+ "  'W' ||
ROUND(ABS(A.BillboardLongitude),3)" sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'E' || ROUND(ABS(A.BillboardLongitude),3)" sSQL=sSQL+ " END ) AS
sBillboardLongitude,"

sSQL=sSQL+ " A.VisitorCountry," sSQL=sSQL+ " A.VisitorPlaceName,"
```

```
sSQL=sSQL+ " ROUND(A.VisitorLatitude,3) AS VisitorLatitude, " sSQL=sSQL+ "
ROUND(A.VisitorLongitude,3) AS VisitorLongitude,"

sSQL=sSQL+ " (CASE WHEN A.VisitorLatitude < 0 THEN "
QL=sSQL+ "  'S' || ROUND(ABS(A.VisitorLatitude),3)" sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'N' ||ROUND(ABS(A.VisitorLatitude),3)" sSQL=sSQL+ " END ) AS
sVisitorLatitude,"

sSQL=sSQL+ " (CASE WHEN A.VisitorLongitude < 0 THEN " sSQL=sSQL+ "  'W' ||
ROUND(ABS(A.VisitorLongitude),3)" sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'E' || ROUND(ABS(A.VisitorLongitude),3)" sSQL=sSQL+ " END ) AS
sVisitorLongitude,"

sSQL=sSQL+ " A.VisitorYearRate" sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_BillboardVistorsData AS A;"
BillboardVistorsData=pd.read_sql_query(sSQL, conn) print('################')
#####################################################################
BillboardVistorsData['Distance']=BillboardVistorsData.apply(lambda row: round(
vincenty((row['BillboardLatitude'],row['BillboardLongitude']),
(row['VisitorLatitude'],row['VisitorLongitude'])).miles
,4)
,axis=1) ######################################################################
G=nx.Graph() ######################################################################

for i in range(BillboardVistorsData.shape[0]):
sNode0='MediaHub-' + BillboardVistorsData['BillboardCountry'][i]
sNode1='B-'+ BillboardVistorsData['sBillboardLatitude'][i] + '-' sNode1=sNode1 +
BillboardVistorsData['sBillboardLongitude'][i] G.add_node(sNode1,
Nodetype='Billboard', Country=BillboardVistorsData['BillboardCountry'][i],
PlaceName=BillboardVistorsData['BillboardPlaceName'][i],
Latitude=round(BillboardVistorsData['BillboardLatitude'][i],3),
Longitude=round(BillboardVistorsData['BillboardLongitude'][i],3))

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

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

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

Output:

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

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

Hence, we have applied formulae to extract features, such as the distance between the billboard and the visitor.

Code:

Planning an Event for Top-Ten Customers

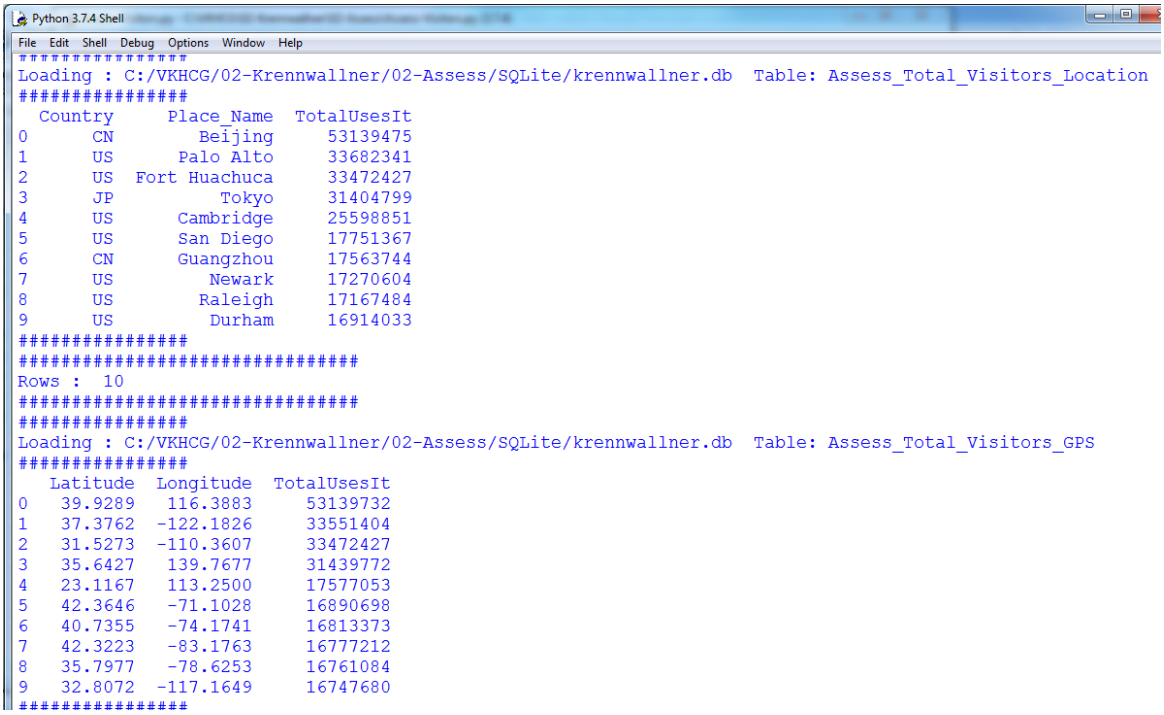
Open Python editor and create a file named Assess-Visitors.py in directory C:\VKHCG\02-Krennwallner\02-Assess

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

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

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " A.Country," sSQL=sSQL+ " A.Place_Name," sSQL=sSQL+ " A.Latitude,"
sSQL=sSQL+ " A.Longitude,"
sSQL=sSQL+ " (A.Last_IP_Number - A.First_IP_Number) AS UsesIt" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor as A" sSQL=sSQL+ " WHERE"
```

```
sSQL=sSQL+ " Country is not null" sSQL=sSQL+ " AND"
sSQL=sSQL+ " Place_Name is not null;" sql.execute(sSQL,conn)
######################################################################
print('#################')
sView='Assess_Total_Visitors_Location' print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " Country," sSQL=sSQL+ " Place_Name,"
sSQL=sSQL+ " SUM(UsesIt) AS TotalUsesIt" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor_UseIt" sSQL=sSQL+ " GROUP BY"
sSQL=sSQL+ " Country," sSQL=sSQL+ " Place_Name" sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " TotalUsesIt DESC" sSQL=sSQL+ " LIMIT 10;"
sql.execute(sSQL,conn)
######################################################################
print('#################')
sView='Assess_Total_Visitors_GPS' print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " Latitude," sSQL=sSQL+ " Longitude,"
sSQL=sSQL+ " SUM(UsesIt) AS TotalUsesIt" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor_UseIt"
sSQL=sSQL+ " GROUP BY"
sSQL=sSQL+ " Latitude," sSQL=sSQL+ " Longitude" sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " TotalUsesIt DESC" sSQL=sSQL+ " LIMIT 10;"
sql.execute(sSQL,conn)
sTables=['Assess_Total_Visitors_Location', 'Assess_Total_Visitors_GPS'] for sTable in sTables:
print('#################')
print('Loading :',sDatabaseName,' Table:',sTable) sSQL=" SELECT "
sSQL=sSQL+ " *" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";" TopData=pd.read_sql_query(sSQL, conn)
print('#################')
print(TopData) print('#################')
print('#################################')
print('Rows : ',TopData.shape[0]) print('#################################')
 print('### Done!! ############################################')
```

Output:



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

Practical 4F: rite a Python / R program to plan the locations of the warehouses from the given data.

Code:

Planning the Locations of the Warehouses
Planning the location of the warehouses requires the assessment of the GPS locations of these warehouses against the requirements for Hillman's logistics needs.
Open your editor and create a file named Assess-Warehouse-Address.py in directory C:\VKHCG\03-Hillman\02-Assess.

```
################### Assess-Warehouse-Address.py  ################### # -*- coding: utf-8 -*- ##################################################################
import os
import pandas as pd
from geopy.geocoders import Nominatim geolocator = Nominatim()
##################################################################
InputDir='01-Retrieve/01-EDS/01-R' InputFileName='Retrieve_GB_Postcode_Warehouse.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python' OutputFileName='Assess_GB_Warehouse_Address.csv'
Company='03-Hillman'
##################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using Windows') print('##################################')
##################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
##################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
##################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName print('############')
print('Loading :',sFileName) Warehouse=pd.read_csv(sFileName,header=0,low_memory=False)
Warehouse.sort_values(by='postcode', ascending=1)
##################################################################
## Limited to 10 due to service limit on Address Service.
##################################################################
```

```
WarehouseGoodHead=Warehouse[Warehouse.latitude != 0].head(5)
WarehouseGoodTail=Warehouse[Warehouse.latitude != 0].tail(5)
#########################################################################
WarehouseGoodHead['Warehouse_Point']=WarehouseGoodHead.apply(lambda row:
(str(row['latitude'])+','+str(row['longitude']))
,axis=1) WarehouseGoodHead['Warehouse_Address']=WarehouseGoodHead.apply(lambda row:
geolocator.reverse(row['Warehouse_Point']).address
axis=1)
WarehouseGoodHead.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodHead.drop('id', axis=1, inplace=True) WarehouseGoodHead.drop('postcode',
axis=1, inplace=True)
WarehouseGoodTail['Warehouse_Point']=WarehouseGoodTail.apply(lambda row:
(str(row['latitude'])+','+str(row['longitude']))
,axis=1) WarehouseGoodTail['Warehouse_Address']=WarehouseGoodTail.apply(lambda row:
geolocator.reverse(row['Warehouse_Point']).address
,axis=1)
WarehouseGoodTail.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodTail.drop('id', axis=1, inplace=True) WarehouseGoodTail.drop('postcode',
axis=1, inplace=True)
WarehouseGood=WarehouseGoodHead.append(WarehouseGoodTail, ignore_index=True)
print(WarehouseGood)
sFileName=sFileDir + '/' + OutputFileName WarehouseGood.to_csv(sFileName, index = False)
print('### Done!! ################################################')
```

Output:

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

Practical 4G: rite a Python / R program using data science via clustering to determine new warehouses using the given data.

Code:

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

Output:

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

Ln: 33 Col: 4

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

Assess_All_Warehouse

Practical No 4H: Using the given data, write a Python / R program to plan the shipping routes for best-fit international logistics.

Code:

```
import sys import os
import pandas as pd import networkx as nx
from geopy.distance import vincenty import sqlite3 as sq
from pandas.io import sql
#####################################################################
if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG' print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
#####################################################################
Company='03-Hillman'
InputDir='01-Retrieve/01-EDS/01-R' InputFileName='Retrieve_All_Countries.csv' EDSDir='02-
Assess/01-EDS' OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_Best_Logistics.gml'
#####################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db' conn = sq.connect(sDatabaseName)
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName print('############')
print('Loading :',sFileName) Warehouse=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
#####################################################################
sColumns={'X1' : 'Country', 'X2' : 'PostCode',
'X3' : 'PlaceName',
'X4' : 'AreaName',
'X5' : 'AreaCode',
'X10' : 'Latitude',
```

```
'X11' : 'Longitude'} Warehouse.rename(columns=sColumns,inplace=True)
WarehouseGood=Warehouse #print(WarehouseGood.head())
#####################################################################
RoutePointsCountry=pd.DataFrame(WarehouseGood.groupby(['Country'])[['Latitude','Longitude
'
]].mean()) #print(RoutePointsCountry.head()) print('#################')
sTable='Assess_RoutePointsCountry' print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsCountry.to_sql(sTable, conn, if_exists="replace") print('#################')
#####################################################################
RoutePointsPostCode=pd.DataFrame(WarehouseGood.groupby(['Country',
'PostCode'])[['Latitude','Longitude']].mean()) #print(RoutePointsPostCode.head())
print('#################')
sTable='Assess_RoutePointsPostCode' print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsPostCode.to_sql(sTable, conn, if_exists="replace") print('#################')
#####################################################################
RoutePointsPlaceName=pd.DataFrame(WarehouseGood.groupby(['Country',
'PostCode','PlaceName'])[['Latitude','Longitude']].mean()) #print(RoutePointsPlaceName.head())
print('#################')
sTable='Assess_RoutePointsPlaceName' print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsPlaceName.to_sql(sTable, conn, if_exists="replace") print('#################')
#####################################################################
### Fit Country to Country
#####################################################################
print('#################')
sView='Assess_RouteCountries'
print('Creating :',sDatabaseName,' View:',sView)
SQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry," sSQL=sSQL+ "   S.Latitude AS
SourceLatitude," sSQL=sSQL+ "   S.Longitude AS SourceLongitude," sSQL=sSQL+ "
T.Country AS TargetCountry," sSQL=sSQL+ "   T.Latitude AS TargetLatitude," sSQL=sSQL+ "
T.Longitude AS TargetLongitude" sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS S" sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS T" sSQL=sSQL+ " WHERE S.Country <>
T.Country" sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')" sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)
```

```
print('#################')
print('Loading :',sDatabaseName,' Table:',sView) sSQL=" SELECT "
sSQL=sSQL+ " *" sSQL=sSQL+ " FROM" sSQL=sSQL+ " " + sView + ";"
RouteCountries=pd.read_sql_query(sSQL, conn)
RouteCountries['Distance']=RouteCountries.apply(lambda row: round(
vincenty((row['SourceLatitude'],row['SourceLongitude']),
(row['TargetLatitude'],row['TargetLongitude'])).miles,4),axis=1)

print(RouteCountries.head(5))
#########################################################################
### Fit Country to Post Code
#########################################################################
print('#################')
sView='Assess_RoutePostCode'
print('Creating :',sDatabaseName,' View:',sView) sSQL="DROP VIEW IF EXISTS " + sView +
";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry,"
sSQL=sSQL+ "   S.Latitude AS SourceLatitude," sSQL=sSQL+ "   S.Longitude AS
SourceLongitude," sSQL=sSQL+ "   T.Country AS TargetCountry," sSQL=sSQL+ "   T.PostCode
AS TargetPostCode," sSQL=sSQL+ "   T.Latitude AS TargetLatitude," sSQL=sSQL+ "
T.Longitude AS TargetLongitude" sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS S" sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsPostCode AS T" sSQL=sSQL+ " WHERE S.Country =
T.Country" sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')" sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)

print('#################')
print('Loading :',sDatabaseName,' Table:',sView) sSQL=" SELECT "
sSQL=sSQL+ " *" sSQL=sSQL+ " FROM" sSQL=sSQL+ " " + sView + ";"
RoutePostCode=pd.read_sql_query(sSQL, conn)

RoutePostCode['Distance']=RoutePostCode.apply(lambda row: round(
vincenty((row['SourceLatitude'],row['SourceLongitude']),
(row['TargetLatitude'],row['TargetLongitude'])).miles
```

```
,4)
,axis=1)
print(RoutePostCode.head(5))
#########################################################################
### Fit Post Code to Place Name
#########################################################################
print('#################')
sView='Assess_RoutePlaceName'   print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry," sSQL=sSQL+ "   S.PostCode AS
SourcePostCode," sSQL=sSQL+ "   S.Latitude AS SourceLatitude," sSQL=sSQL+ "
S.Longitude AS SourceLongitude,"
sSQL=sSQL+ "   T.Country AS TargetCountry," sSQL=sSQL+ "   T.PostCode AS
TargetPostCode," sSQL=sSQL+ "   T.PlaceName AS TargetPlaceName," sSQL=sSQL+ "
T.Latitude AS TargetLatitude," sSQL=sSQL+ "   T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsPostCode AS S" sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsPLaceName AS T" sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " S.Country = T.Country" sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.PostCode = T.PostCode" sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')" sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)

print('#################')
print('Loading :',sDatabaseName,' Table:',sView) sSQL=" SELECT "
sSQL=sSQL+ " *" sSQL=sSQL+ " FROM" sSQL=sSQL+ " " + sView + ";"
RoutePlaceName=pd.read_sql_query(sSQL, conn)

RoutePlaceName['Distance']=RoutePlaceName.apply(lambda row: round(
vincenty((row['SourceLatitude'],row['SourceLongitude']),
(row['TargetLatitude'],row['TargetLongitude'])).miles
,4)
,axis=1)

print(RoutePlaceName.head(5))
#########################################################################
```

```
G=nx.Graph() ######################################################################
print('Countries:',RouteCountries.shape) for i in range(RouteCountries.shape[0]):
sNode0='C-' + RouteCountries['SourceCountry'][i] G.add_node(sNode0,
Nodetype='Country', Country=RouteCountries['SourceCountry'][i],
Latitude=round(RouteCountries['SourceLatitude'][i],4),
Longitude=round(RouteCountries['SourceLongitude'][i],4))

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

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

```
sNode1='L-' + RoutePlaceName['TargetPlaceName'][i]  + '-' sNode1=sNode1 +
RoutePlaceName['TargetPostCode'][i]  + '-' sNode1=sNode1 +
RoutePlaceName['TargetCountry'][i] G.add_node(sNode1,
Nodetype='PlaceName', Country=RoutePlaceName['TargetCountry'][i],
PostCode=RoutePlaceName['TargetPostCode'][i],
PlaceName=RoutePlaceName['TargetPlaceName'][i],
Latitude=round(RoutePlaceName['TargetLatitude'][i],4),
Longitude=round(RoutePlaceName['TargetLongitude'][i],4))
G.add_edge(sNode0,sNode1,distance=round(RoutePlaceName['Distance'][i],3))
#print(sNode0,sNode1)
#####################################################################
sFileName=sFileDir + '/' + OutputFileName print('#################################')
print('Storing :', sFileName) print('#################################')
nx.write_gml(G,sFileName) sFileName=sFileName +'.gz' nx.write_gml(G,sFileName)
#####################################################################
print('#################################')
print('Path:', nx.shortest_path(G,source='P-SW1-GB',target='P-01001-US',weight='distance'))
print('Path length:', nx.shortest_path_length(G,source='P-SW1-GB',target='P-01001-
US',weight='distance'))
print('Path length (1):', nx.shortest_path_length(G,source='P-SW1-GB',target='C-
GB',weight='distance'))
print('Path length (2):', nx.shortest_path_length(G,source='C-GB',target='C-
US',weight='distance'))
print('Path length (3):', nx.shortest_path_length(G,source='C-US',target='P-01001-
US',weight='distance'))
print('#################################')
print('Routes from P-SW1-GB < 2: ', nx.single_source_shortest_path(G,source='P-SW1-GB'
,cutoff=1))
print('Routes from P-01001-US < 2: ', nx.single_source_shortest_path(G,source='P-01001-US'
,cutoff=1)) print('#################################')
#####################################################################
print('################')
print('Vacuum Database') sSQL="VACUUM;"
sql.execute(sSQL,conn) print('################')
##################################################################### print('###
Done!! ############################################')
```

Output:

You can now query features out of a graph, such as shortage paths between locations and paths from a given location, using Assess_Best_Logistics.gml with appropirate application.

Practical No 4I: Write a Python / R program to decide the best packing option to ship in container from the given data.

Code:

```
import sys import os
import pandas as pd import sqlite3 as sq
from pandas.io import sql
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
Company='03-Hillman'
InputDir='01-Retrieve/01-EDS/02-Python' InputFileName1='Retrieve_Product.csv'
InputFileName2='Retrieve_Box.csv' InputFileName3='Retrieve_Container.csv' EDSDir='02-
Assess/01-EDS' OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_Shipping_Containers.csv'
#####################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/hillman.db' conn = sq.connect(sDatabaseName)
### Import Product Data
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName1 print('###########')
print('Loading :',sFileName) ProductRawData=pd.read_csv(sFileName,
header=0, low_memory=False, encoding="latin-1"
)
```

```
ProductRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ProductRawData.index.name = 'IDNumber'
ProductData=ProductRawData[ProductRawData.Length <= 0.5].head(10) print('Loaded Product
:',ProductData.columns.values) print('##################################')
#####################################################################
print('#################')
sTable='Assess_Product'
print('Storing :',sDatabaseName,' Table:',sTable) ProductData.to_sql(sTable, conn,
if_exists="replace") print('#################')
#####################################################################
print(ProductData.head()) print('##################################')
print('Rows : ',ProductData.shape[0]) print('##################################')
#####################################################################
#####################################################################
### Import Box Data
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName2 print('###########')
print('Loading :',sFileName) BoxRawData=pd.read_csv(sFileName,
header=0, low_memory=False, encoding="latin-1"
)
BoxRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BoxRawData.index.name = 'IDNumber' BoxData=BoxRawData[BoxRawData.Length <=
1].head(1000) print('Loaded Product :',BoxData.columns.values)
print('##################################')
#####################################################################
print('#################')
sTable='Assess_Box'
print('Storing :',sDatabaseName,' Table:',sTable)
BoxData.to_sql(sTable, conn, if_exists="replace") print('#################')
#####################################################################
print(BoxData.head()) print('##################################')
print('Rows : ',BoxData.shape[0]) print('##################################')
#####################################################################
#####################################################################
### Import Container Data
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName3 print('###########')
print('Loading :',sFileName) ContainerRawData=pd.read_csv(sFileName,
header=0, low_memory=False, encoding="latin-1"
)
```

```
ContainerRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ContainerRawData.index.name = 'IDNumber'
ContainerData=ContainerRawData[ContainerRawData.Length <= 2].head(10) print('Loaded
Product :',ContainerData.columns.values) print('##################################')
#####################################################################
print('#################')
sTable='Assess_Container'
print('Storing :',sDatabaseName,' Table:',sTable) BoxData.to_sql(sTable, conn,
if_exists="replace") print('#################')
#####################################################################
print(ContainerData.head()) print('##################################')
print('Rows : ',ContainerData.shape[0]) print('##################################')
#####################################################################
#####################################################################
### Fit Product in Box
#####################################################################
print('#################')
sView='Assess_Product_in_Box'
print('Creating :',sDatabaseName,' View:',sView) sSQL="DROP VIEW IF EXISTS " + sView +
";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " P.UnitNumber AS ProductNumber,"
sSQL=sSQL+ " B.UnitNumber AS BoxNumber," sSQL=sSQL+ " (B.Thickness * 1000) AS
PackSafeCode,"
sSQL=sSQL+ " (B.BoxVolume - P.ProductVolume) AS PackFoamVolume," sSQL=sSQL+ "
((B.Length*10) * (B.Width*10) * (B.Height*10)) * 167 AS Air_Dimensional_Weight,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 333 AS
Road_Dimensional_Weight,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 1000 AS
Sea_Dimensional_Weight,"
sSQL=sSQL+ " P.Length AS Product_Length," sSQL=sSQL+ " P.Width AS Product_Width,"
sSQL=sSQL+ " P.Height AS Product_Height,"
sSQL=sSQL+ " P.ProductVolume AS Product_cm_Volume,"
sSQL=sSQL+ " ((P.Length*10) * (P.Width*10) * (P.Height*10)) AS Product_ccm_Volume,"
sSQL=sSQL+ " (B.Thickness * 0.95) AS Minimum_Pack_Foam,"
sSQL=sSQL+ " (B.Thickness * 1.05) AS Maximum_Pack_Foam,"
```

```
sSQL=sSQL+ " B.Length - (B.Thickness * 1.10) AS Minimum_Product_Box_Length,"
sSQL=sSQL+ " B.Length - (B.Thickness * 0.95) AS Maximum_Product_Box_Length,"
sSQL=sSQL+ " B.Width - (B.Thickness * 1.10) AS Minimum_Product_Box_Width,"
sSQL=sSQL+ " B.Width - (B.Thickness * 0.95) AS Maximum_Product_Box_Width,"
sSQL=sSQL+ " B.Height - (B.Thickness * 1.10) AS Minimum_Product_Box_Height,"
sSQL=sSQL+ " B.Height - (B.Thickness * 0.95) AS Maximum_Product_Box_Height,"
sSQL=sSQL+ " B.Length AS Box_Length,"
sSQL=sSQL+ " B.Width AS Box_Width," sSQL=sSQL+ " B.Height AS Box_Height,"
sSQL=sSQL+ " B.BoxVolume AS Box_cm_Volume,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) AS Box_ccm_Volume,"
sSQL=sSQL+ " (2 * B.Length * B.Width) + (2 * B.Length * B.Height) + (2 * B.Width *
B.Height) AS Box_sqm_Area,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) *  3.5 AS
Box_A_Max_Kg_Weight,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) *  7.7 AS
Box_B_Max_Kg_Weight,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 10.0 AS
Box_C_Max_Kg_Weight"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Product as P" sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Box as B" sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " P.Length >= (B.Length - (B.Thickness * 1.10))" sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Width >= (B.Width - (B.Thickness * 1.10))" sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Height >= (B.Height - (B.Thickness * 1.10))" sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Length <= (B.Length - (B.Thickness * 0.95))" sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Width <= (B.Width - (B.Thickness * 0.95))" sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Height <= (B.Height - (B.Thickness * 0.95))" sSQL=sSQL+ " AND"
sSQL=sSQL+ " (B.Height - B.Thickness) >= 0" sSQL=sSQL+ " AND"
sSQL=sSQL+ " (B.Width - B.Thickness) >= 0" sSQL=sSQL+ " AND"
sSQL=sSQL+ " (B.Height - B.Thickness) >= 0" sSQL=sSQL+ " AND"
sSQL=sSQL+ " B.BoxVolume >= P.ProductVolume;" sql.execute(sSQL,conn)
#########################################################################
### Fit Box in Pallet
#########################################################################
t=0
for l in range(2,8):
for w in range(2,8): for h in range(4):
t += 1
PalletLine=[('IDNumber',[t]),
('ShipType', ['Pallet']),
```

```
('UnitNumber', ('L-'+format(t,"06d"))),
('Box_per_Length',(format(2**l,"4d"))),
('Box_per_Width',(format(2**w,"4d"))),
('Box_per_Height',(format(2**h,"4d")))] if t==1:
PalletFrame = pd.DataFrame.from_items(PalletLine) else:
PalletRow = pd.DataFrame.from_items(PalletLine) PalletFrame =
PalletFrame.append(PalletRow)
PalletFrame.set_index(['IDNumber'],inplace=True)
PalletFrame.head() print('###################################')
print('Rows : ',PalletFrame.shape[0]) print('###################################')
### Fit Box on Pallet print('#################')
sView='Assess_Box_on_Pallet'
print('Creating :',sDatabaseName,' View:',sView) sSQL="DROP VIEW IF EXISTS " + sView +
";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " P.UnitNumber AS PalletNumber," sSQL=sSQL+ " B.UnitNumber AS
BoxNumber,"
sSQL=sSQL+ " round(B.Length*P.Box_per_Length,3) AS Pallet_Length," sSQL=sSQL+ "
round(B.Width*P.Box_per_Width,3) AS Pallet_Width," sSQL=sSQL+ "
round(B.Height*P.Box_per_Height,3) AS Pallet_Height,"
sSQL=sSQL+ " P.Box_per_Length * P.Box_per_Width * P.Box_per_Height AS Pallet_Boxes"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Box as B" sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Pallet as P" sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " round(B.Length*P.Box_per_Length,3) <= 20" sSQL=sSQL+ " AND"
sSQL=sSQL+ " round(B.Width*P.Box_per_Width,3) <= 9" sSQL=sSQL+ " AND"
sSQL=sSQL+ " round(B.Height*P.Box_per_Height,3) <= 5;" sql.execute(sSQL,conn)
sTables=['Assess_Product_in_Box','Assess_Box_on_Pallet'] for sTable in sTables:
print('#################')
print('Loading :',sDatabaseName,' Table:',sTable) sSQL=" SELECT "
sSQL=sSQL+ " *" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";" SnapShotData=pd.read_sql_query(sSQL, conn)
print('#################')
sTableOut=sTable + '_SnapShot'
print('Storing :',sDatabaseName,' Table:',sTable) SnapShotData.to_sql(sTableOut, conn,
if_exists="replace") print('#################')
### Fit Pallet in Container
sTables=['Length','Width','Height'] for sTable in sTables:
```

```
sView='Assess_Pallet_in_Container_' + sTable print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " C.UnitNumber AS ContainerNumber," sSQL=sSQL+ " P.PalletNumber,"
sSQL=sSQL+ " P.BoxNumber,"
sSQL=sSQL+ " round(C." + sTable + "/P.Pallet_" + sTable + ",0)" sSQL=sSQL+ " AS
Pallet_per_" + sTable + ","
sSQL=sSQL+ " round(C." + sTable + "/P.Pallet_" + sTable + ",0)" sSQL=sSQL+ " *
P.Pallet_Boxes AS Pallet_" + sTable + "_Boxes," sSQL=sSQL+ " P.Pallet_Boxes"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Container as C" sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Box_on_Pallet_SnapShot as P" sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " round(C.Length/P.Pallet_Length,0) > 0" sSQL=sSQL+ " AND"
sSQL=sSQL+ " round(C.Width/P.Pallet_Width,0) > 0" sSQL=sSQL+ " AND"
sSQL=sSQL+ " round(C.Height/P.Pallet_Height,0) > 0;" sql.execute(sSQL,conn)
print('#################')
print('Loading :',sDatabaseName,' Table:',sView) sSQL=" SELECT "
sSQL=sSQL+ " *" sSQL=sSQL+ " FROM" sSQL=sSQL+ " " + sView + ";"
SnapShotData=pd.read_sql_query(sSQL, conn) print('#################')
sTableOut= sView + '_SnapShot'
print('Storing :',sDatabaseName,' Table:',sTableOut) SnapShotData.to_sql(sTableOut, conn,
if_exists="replace") print('#################')
#####################################################################
print('#################')
sView='Assess_Pallet_in_Container' print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " CL.ContainerNumber," sSQL=sSQL+ " CL.PalletNumber," sSQL=sSQL+ "
CL.BoxNumber,"
sSQL=sSQL+ " CL.Pallet_Boxes AS Boxes_per_Pallet," sSQL=sSQL+ "
CL.Pallet_per_Length,"
sSQL=sSQL+ " CW.Pallet_per_Width," sSQL=sSQL+ " CH.Pallet_per_Height,"
sSQL=sSQL+ " CL.Pallet_Length_Boxes * CW.Pallet_Width_Boxes * CH.Pallet_Height_Boxes
AS Container_Boxes"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Pallet_in_Container_Length_SnapShot as CL" sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " Assess_Pallet_in_Container_Width_SnapShot as CW" sSQL=sSQL+ " ON"
sSQL=sSQL+ " CL.ContainerNumber = CW.ContainerNumber" sSQL=sSQL+ " AND"
```

```
sSQL=sSQL+ " CL.PalletNumber = CW.PalletNumber" sSQL=sSQL+ " AND"
sSQL=sSQL+ " CL.BoxNumber = CW.BoxNumber" sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " Assess_Pallet_in_Container_Height_SnapShot as CH" sSQL=sSQL+ " ON"
sSQL=sSQL+ " CL.ContainerNumber = CH.ContainerNumber" sSQL=sSQL+ " AND"
sSQL=sSQL+ " CL.PalletNumber = CH.PalletNumber" sSQL=sSQL+ " AND"
sSQL=sSQL+ " CL.BoxNumber = CH.BoxNumber;" sql.execute(sSQL,conn)
#########################################################################
sTables=['Assess_Product_in_Box','Assess_Pallet_in_Container'] for sTable in sTables:
print('##################')
print('Loading :',sDatabaseName,' Table:',sTable) sSQL=" SELECT "
sSQL=sSQL+ " *" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";" PackData=pd.read_sql_query(sSQL, conn)
print('##################')
print(PackData) print('##################')
print('###################################')
print('Rows : ',PackData.shape[0]) print('###################################')
sFileName=sFileDir + '/' + sTable + '.csv' print(sFileName) PackData.to_csv(sFileName, index =
False)
print('### Done!! ################################################')
```

Output:

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:\VKHCG\03-Hillman\02-Assess\Assess-Shipping-Containers.py =====
################################
Working Base : C:/VKHCG  using  win32
################################
############
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/02-Python/Retrieve_Product.csv
Loaded Product : ['ShipType' 'UnitNumber' 'Length' 'Width' 'Height' 'ProductVolu
me']
################################
#################
Storing : C:/VKHCG/03-Hillman/02-Assess/SQLite/hillman.db  Table: Assess_Product
#################
        ShipType UnitNumber  Length  Width  Height  ProductVolume
IDNumber
0         Product    P000001     0.1    0.1     0.1          0.001
1         Product    P000002     0.1    0.1     0.2          0.002
2         Product    P000003     0.1    0.1     0.3          0.003
3         Product    P000004     0.1    0.1     0.4          0.004
4         Product    P000005     0.1    0.1     0.5          0.005
################################
Rows :  10
################################
############
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/02-Python/Retrieve_Box.csv
Loaded Product : ['ShipType' 'UnitNumber' 'Length' 'Width' 'Height' 'Thickness'
'BoxVolume'
 'ProductVolume']
################################
#################
```

Practical No 4J: Write a Python program to create a delivery route using the given data.

Code:

```
import sys import os
import pandas as pd import sqlite3 as sq
from pandas.io import sql import networkx as nx
from geopy.distance import vincenty
#####################################################################
nMax=3 nMaxPath=10 nSet=False nVSet=False
#####################################################################
Base='C:/VKHCG'
#####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
Company='03-Hillman'
InputDir1='01-Retrieve/01-EDS/01-R' InputDir2='01-Retrieve/01-EDS/02-Python'
InputFileName1='Retrieve_GB_Postcode_Warehouse.csv'
InputFileName2='Retrieve_GB_Postcodes_Shops.csv' EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python' OutputFileName1='Assess_Shipping_Routes.gml'
OutputFileName2='Assess_Shipping_Routes.txt'
#####################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/hillman.db' conn = sq.connect(sDatabaseName)
#####################################################################
#####################################################################
### Import Warehouse Data
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir1 + '/' + InputFileName1 print('############')
```

```
print('Loading :',sFileName) WarehouseRawData=pd.read_csv(sFileName,
header=0, low_memory=False, encoding="latin-1"
)
WarehouseRawData.drop_duplicates(subset=None, keep='first', inplace=True)
WarehouseRawData.index.name = 'IDNumber' WarehouseData=WarehouseRawData.head(nMax)
WarehouseData=WarehouseData.append(WarehouseRawData.tail(nMax))
WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode=='
KA13'])  if nSet==True:

WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode=='S
W1W']) WarehouseData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Loaded Warehouses :',WarehouseData.columns.values)
print('##################################')
#####################################################################
print('#################')
sTable='Assess_Warehouse_UK'
print('Storing :',sDatabaseName,' Table:',sTable) WarehouseData.to_sql(sTable, conn,
if_exists="replace") print('#################')
#####################################################################
print(WarehouseData.head()) print('##################################')
print('Rows : ',WarehouseData.shape[0]) print('##################################')
#####################################################################
### Import Shop Data
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir1 + '/' + InputFileName2 print('###########')
print('Loading :',sFileName) ShopRawData=pd.read_csv(sFileName,
header=0, low_memory=False, encoding="latin-1"
)
ShopRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ShopRawData.index.name = 'IDNumber'
ShopData=ShopRawData
print('Loaded Shops :',ShopData.columns.values)
print('##################################')
#####################################################################
print('#################')
sTable='Assess_Shop_UK'
print('Storing :',sDatabaseName,' Table:',sTable) ShopData.to_sql(sTable, conn,
if_exists="replace") print('#################')
#####################################################################
print(ShopData.head()) print('##################################')
```

```
print('Rows : ',ShopData.shape[0]) print('#################################')
#####################################################################
### Connect HQ
#####################################################################
print('#################')
sView='Assess_HQ'
print('Creating :',sDatabaseName,' View:',sView) sSQL="DROP VIEW IF EXISTS " + sView +
";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " W.postcode AS HQ_PostCode," sSQL=sSQL+ " 'HQ-' || W.postcode AS
HQ_Name," sSQL=sSQL+ " round(W.latitude,6) AS HQ_Latitude," sSQL=sSQL+ "
round(W.longitude,6) AS HQ_Longitude" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Warehouse_UK as W" sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " TRIM(W.postcode) in ('KA13','SW1W');"
sql.execute(sSQL,conn)
#####################################################################
### Connect Warehouses
#####################################################################
print('#################')
sView='Assess_Warehouse'
print('Creating :',sDatabaseName,' View:',sView) sSQL="DROP VIEW IF EXISTS " + sView +
";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " W.postcode AS Warehouse_PostCode," sSQL=sSQL+ " 'WH-' || W.postcode AS
Warehouse_Name," sSQL=sSQL+ " round(W.latitude,6) AS Warehouse_Latitude,"
sSQL=sSQL+ " round(W.longitude,6) AS Warehouse_Longitude" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Warehouse_UK as W;" sql.execute(sSQL,conn)
print('#################')
sView='Assess_Shop'
print('Creating :',sDatabaseName,' View:',sView) sSQL="DROP VIEW IF EXISTS " + sView +
";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS" sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " TRIM(S.postcode) AS Shop_PostCode,"
```

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

```
sTable = 'Assess_Shop'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";" RouteData=pd.read_sql_query(sSQL, conn)
print('#################')
print(RouteData.head()) print('###################################')
print('Shop Rows : ',RouteData.shape[0]) print('###################################')
for i in range(RouteData.shape[0]): sNode0=RouteData['Shop_Name'][i] G.add_node(sNode0,
Nodetype='Shop',  PostCode=RouteData['Shop_PostCode'][i],
WarehousePostCode=RouteData['Warehouse_PostCode'][i],
Latitude=round(RouteData['Shop_Latitude'][i],6),
Longitude=round(RouteData['Shop_Longitude'][i],6))
#####################################################################
## Create Edges ###################################################################
print('###################################')
print('Loading Edges') print('###################################')
for sNode0 in nx.nodes_iter(G): for sNode1 in nx.nodes_iter(G):
if G.node[sNode0]['Nodetype']=='HQ' and \ G.node[sNode1]['Nodetype']=='HQ' and \ sNode0 !=
sNode1:
distancemeters=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).meters\

distancemiles=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).miles\
,3)
if distancemiles >= 0.05:
```

```
cost = round(150+(distancemiles * 2.5),6) vehicle='V001'
else:
cost = round(2+(distancemiles * 0.10),6) vehicle='ForkLift'
G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \ DistanceMiles=distancemiles, \
Cost=cost,Vehicle=vehicle)
if nVSet==True:
print('Edge-H-H:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\
distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
if G.node[sNode0]['Nodetype']=='HQ' and \ G.node[sNode1]['Nodetype']=='Warehouse' and \
sNode0 != sNode1:
distancemeters=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).meters\
,0)
distancemiles=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).miles\
,3)
if distancemiles >= 10:
cost = round(50+(distancemiles * 2),6) vehicle='V002'
else:
cost = round(5+(distancemiles * 1.5),6) vehicle='V003'
if distancemiles <= 50: G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
DistanceMiles=distancemiles, \ Cost=cost,Vehicle=vehicle)
if nVSet==True:
print('Edge-H-W:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\
distancemiles,'miles','Cost',  cost,'Vehicle',vehicle)
```

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

if G.node[sNode0]['Nodetype']=='Warehouse' and \ G.node[sNode1]['Nodetype']=='Shop' and \
G.node[sNode0]['PostCode']==G.node[sNode1]['WarehousePostCode'] and \ sNode0 != sNode1:

distancemeters=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
```

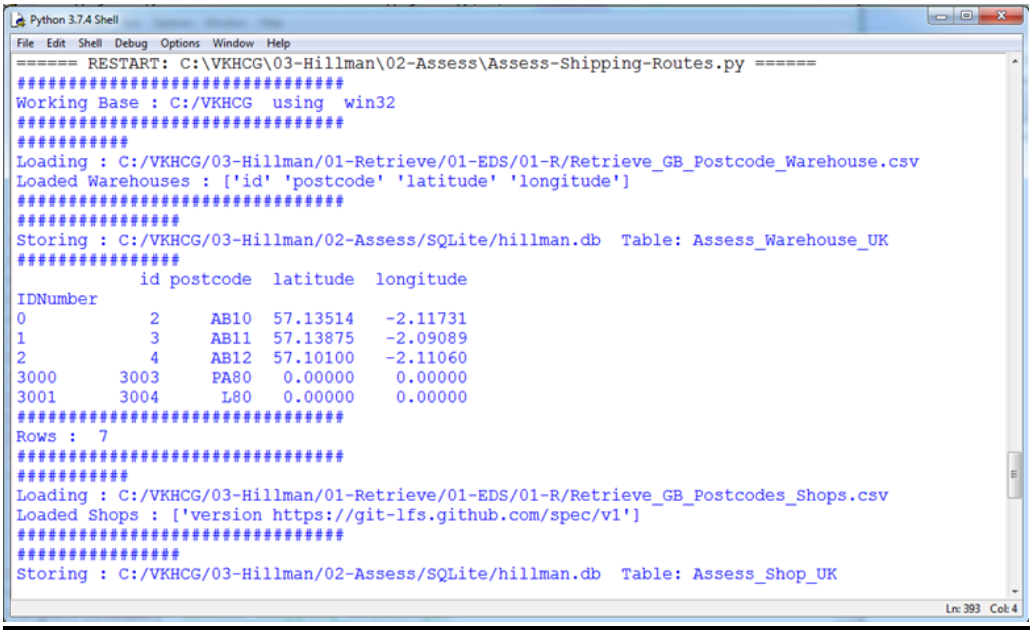
```
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).meters\
,0)
distancemiles=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).miles\
,3)
if distancemiles >= 10:
cost = round(50+(distancemiles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

if nSet==True and \ G.node[sNode0]['Nodetype']=='Shop' and \
G.node[sNode1]['Nodetype']=='Shop' and \
G.node[sNode0]['WarehousePostCode']==G.node[sNode1]['WarehousePostCode'] and \ sNode0
!= sNode1:

distancemeters=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).meters\
,0)
```

```
distancemiles=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).miles\
,3)

if distancemiles >= 0.05:
cost = round(5+(distancemiles * 0.5),6) vehicle='V008'
else:
cost = round(1+(distancemiles * 0.1),6) vehicle='V009'

if distancemiles <= 0.075: G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
DistanceMiles=distancemiles, \ Cost=cost,Vehicle=vehicle)
if nVSet==True:
print('Edge-S-S:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\
distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
if nSet==True and \ G.node[sNode0]['Nodetype']=='Shop' and \
G.node[sNode1]['Nodetype']=='Shop' and \
G.node[sNode0]['WarehousePostCode']!=G.node[sNode1]['WarehousePostCode'] and \ sNode0
!= sNode1:
distancemeters=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).meters\
,0)
distancemiles=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
```

```
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).miles\
,3)

cost = round(1+(distancemiles * 0.1),6) vehicle='V010'
if distancemiles <= 0.025: G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
DistanceMiles=distancemiles, \ Cost=cost,Vehicle=vehicle)
if nVSet==True:
print('Edge-S-S:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\

if distancemiles <= 0.075: G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
DistanceMiles=distancemiles, \ Cost=cost,Vehicle=vehicle)
if nVSet==True:
print('Edge-S-S:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\
distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
if nSet==True and \ G.node[sNode0]['Nodetype']=='Shop' and \
G.node[sNode1]['Nodetype']=='Shop' and \
G.node[sNode0]['WarehousePostCode']!=G.node[sNode1]['WarehousePostCode'] and \ sNode0
!= sNode1:
distancemeters=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).meters\
,0)
distancemiles=round(\
vincenty(\
(\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\
),\ (\
G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\
)\
).miles\
```

```
,3)

cost = round(1+(distancemiles * 0.1),6) vehicle='V010'
if distancemiles <= 0.025: G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
DistanceMiles=distancemiles, \ Cost=cost,Vehicle=vehicle)
if nVSet==True:
print('Edge-S-S:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\
nx.shortest_path_length(G, \ source=sNode0, \ target=sNode1, \ weight='DistanceMeters'))
sline = sID + '|"DistanceMeters"|"' + sNode0 + '"|"' \
+ sNode1 + '"|"' + spath + '"|' + slength if nVSet==True: print (sline) f.write(sline + '\n')
l+=1
sID='{:.0f}'.format(l)
spath = ','.join(nx.shortest_path(G, \ source=sNode0, \
target=sNode1, \ weight='Cost'))
slength= '{:.6f}'.format(\ nx.shortest_path_length(G, \ source=sNode0, \ target=sNode1, \
weight='Cost'))
sline = sID + '|"Cost"|"' + sNode0 + '"|"' \
+ sNode1 + '"|"' + spath + '"|' + slength if nVSet==True: print (sline) f.write(sline + '\n')

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

Output:


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
====== RESTART: C:\VKHCG\03-Hillman\02-Assess\Assess-Shipping-Routes.py ======
################################
Working Base : C:/VKHCG using win32
################################
###########
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_GB_Postcode_Warehouse.csv
Loaded Warehouses : ['id' 'postcode' 'latitude' 'longitude']
################################
################
Storing : C:/VKHCG/03-Hillman/02-Assess/SQLite/hillman.db Table: Assess_Warehouse_UK
################
id postcode latitude longitude
IDNumber
0 2 AB10 57.13514 -2.11731
1 3 AB11 57.13875 -2.09089
2 4 AB12 57.10100 -2.11060
3000 3003 PA80 0.00000 0.00000
3001 3004 L80 0.00000 0.00000
################################
Rows : 7
################################
###########
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_GB_Postcodes_Shops.csv
Loaded Shops : ['version https://git-lfs.github.com/spec/v1']
################################
################
Storing : C:/VKHCG/03-Hillman/02-Assess/SQLite/hillman.db Table: Assess_Shop_UK
Ln: 393 Col: 4

Practical No 4K: Write a Python program to create Simple forex trading planner from the given data.

Code:

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

```
print('Rows : ',CountryData.shape[0]) print('#################################')
#####################################################################
### Import Forex Data
#####################################################################
sFileName2=Base + '/' + sInputFileName2 print('#################################')
print('Loading :',sFileName2) print('#################################')
ForexRawData=pd.read_csv(sFileName2,header=0,low_memory=False, encoding="latin-1")
ForexRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ForexData=ForexRawData.head(5)
print('Loaded Company :',ForexData.columns.values)
print('#################################')
#####################################################################
print('#################')
sTable='Assess_Forex'
print('Storing :',sDatabaseName,' Table:',sTable) ForexData.to_sql(sTable, conn,
if_exists="replace") print('#################')
#####################################################################
print(ForexData.head()) print('#################################')
print('Rows : ',ForexData.shape[0]) print('#################################')
#####################################################################
print('#################')
sTable='Assess_Forex'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="select distinct"
sSQL=sSQL+ " A.CodeIn" sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_Forex as A;" CodeData=pd.read_sql_query(sSQL, conn)
print('#################')
#####################################################################

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

```
sSQL=sSQL+ " Assess_Country as C" sSQL=sSQL+ " ON A.CodeOut = C.CurrencyCode"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " A.CodeIn ='" + CodeData['CodeIn'][c] + "';"
ForexData=pd.read_sql_query(sSQL, conn).head(1000) print('#################')
print(ForexData) print('#################')
sTable='Assess_Forex_' + CodeData['CodeIn'][c] print('Storing :',sDatabaseName,'
Table:',sTable) ForexData.to_sql(sTable, conn, if_exists="replace") print('#################')
print('###################################')
print('Rows : ',ForexData.shape[0]) print('###################################')
######################################################################## print('###
Done!! ##############################################')
######################################################################
```

Output:

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

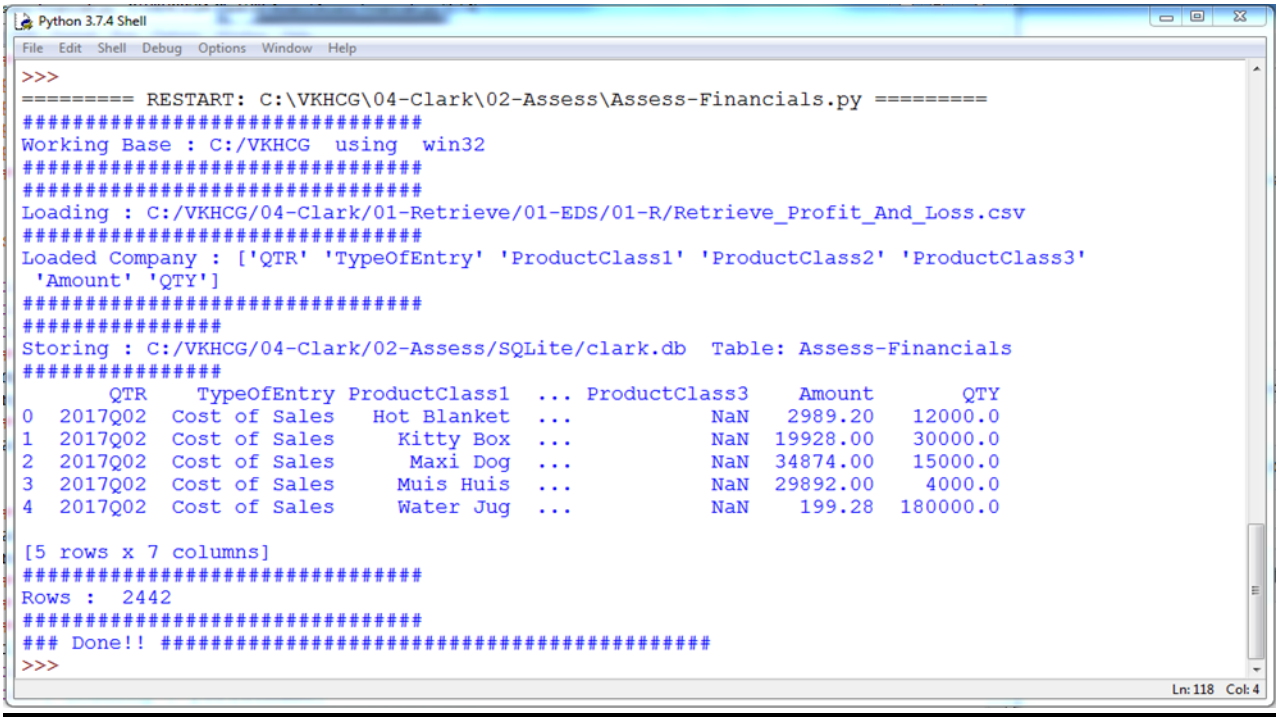
Practical No 4L: Write a Python program to process the balance sheet to ensure that only good data is processing.

Code:

```
import sys import os
import sqlite3 as sq import pandas as pd
#####################################################################
if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG' print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
Company='04-Clark'
sInputFileName='01-Retrieve/01-EDS/01-R/Retrieve_Profit_And_Loss.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/clark.db' conn = sq.connect(sDatabaseName)
#####################################################################
### Import Financial Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('###################################')
print('Loading :',sFileName) print('###################################')
FinancialRawData=pd.read_csv(sFileName,header=0,low_memory=False,  encoding="latin-1")
FinancialData=FinancialRawData
print('Loaded Company :',FinancialData.columns.values)
print('###################################')
#####################################################################
print('#################')
sTable='Assess-Financials'
print('Storing :',sDatabaseName,' Table:',sTable) FinancialData.to_sql(sTable, conn,
if_exists="replace") print('#################')
#####################################################################
print(FinancialData.head()) print('###################################')
```

print('Rows : ',FinancialData.shape[0]) print('##################################')
##
print('###
Done!! ##')

Output:



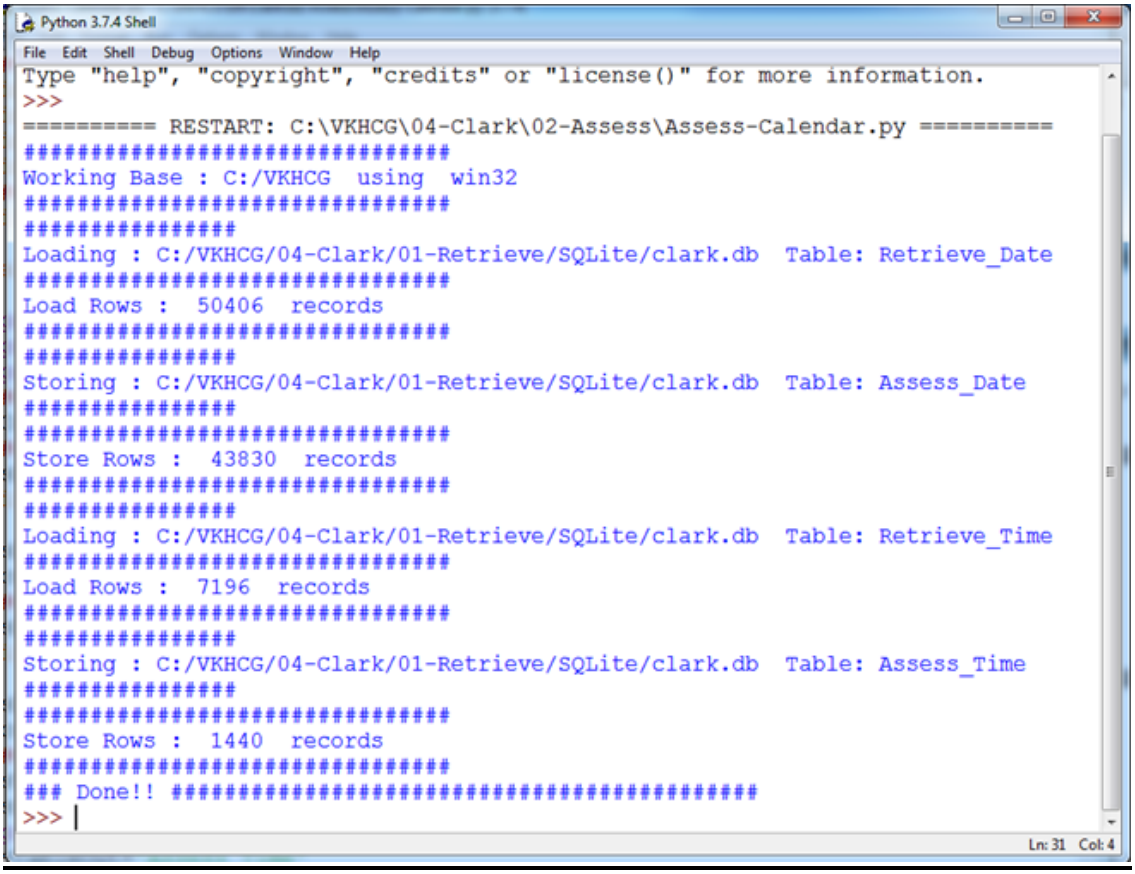
Practical No 4M: Write a Python program to store all master records for the financial calendar

Code:

```
import sys import os
import sqlite3 as sq import pandas as pd
#####################################################################
if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG' print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
Company='04-Clark'
#####################################################################
sDataBaseDirIn=Base + '/' + Company + '/01-Retrieve/SQLite' if not
os.path.exists(sDataBaseDirIn):
os.makedirs(sDataBaseDirIn) sDatabaseNameIn=sDataBaseDirIn + '/clark.db' connIn =
sq.connect(sDatabaseNameIn)
#####################################################################
sDataBaseDirOut=Base + '/' + Company + '/01-Retrieve/SQLite' if not
os.path.exists(sDataBaseDirOut):
os.makedirs(sDataBaseDirOut) sDatabaseNameOut=sDataBaseDirOut + '/clark.db' connOut =
sq.connect(sDatabaseNameOut)
#####################################################################
sTableIn='Retrieve_Date'
sSQL='select * FROM ' + sTableIn + ';' print('#################')
sTableOut='Assess_Time'
print('Loading :',sDatabaseNameIn,' Table:',sTableIn) dateRawData=pd.read_sql_query(sSQL,
connIn) dateData=dateRawData
#####################################################################
print('##################################')
print('Load Rows : ',dateRawData.shape[0], ' records')
print('##################################')
dateData.drop_duplicates(subset='FinDate', keep='first', inplace=True)
#####################################################################
print('#################')
sTableOut='Assess_Date'
print('Storing :',sDatabaseNameOut,' Table:',sTableOut)
```

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

Output:


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\VKHCG\04-Clark\02-Assess\Assess-Calendar.py ==========
################################
Working Base : C:/VKHCG using win32
################################
################
Loading : C:/VKHCG/04-Clark/01-Retrieve/SQLite/clark.db Table: Retrieve_Date
################################
Load Rows : 50406 records
################################
################
Storing : C:/VKHCG/04-Clark/01-Retrieve/SQLite/clark.db Table: Assess_Date
################
################################
Store Rows : 43830 records
################################
################
Loading : C:/VKHCG/04-Clark/01-Retrieve/SQLite/clark.db Table: Retrieve_Time
################################
Load Rows : 7196 records
################################
################
Storing : C:/VKHCG/04-Clark/01-Retrieve/SQLite/clark.db Table: Assess_Time
################
################################
Store Rows : 1440 records
################################
Done!!
>>>
Ln: 31 Col: 4

Practical No 4N: Write a Python program to generate payroll from the given data.

Code:

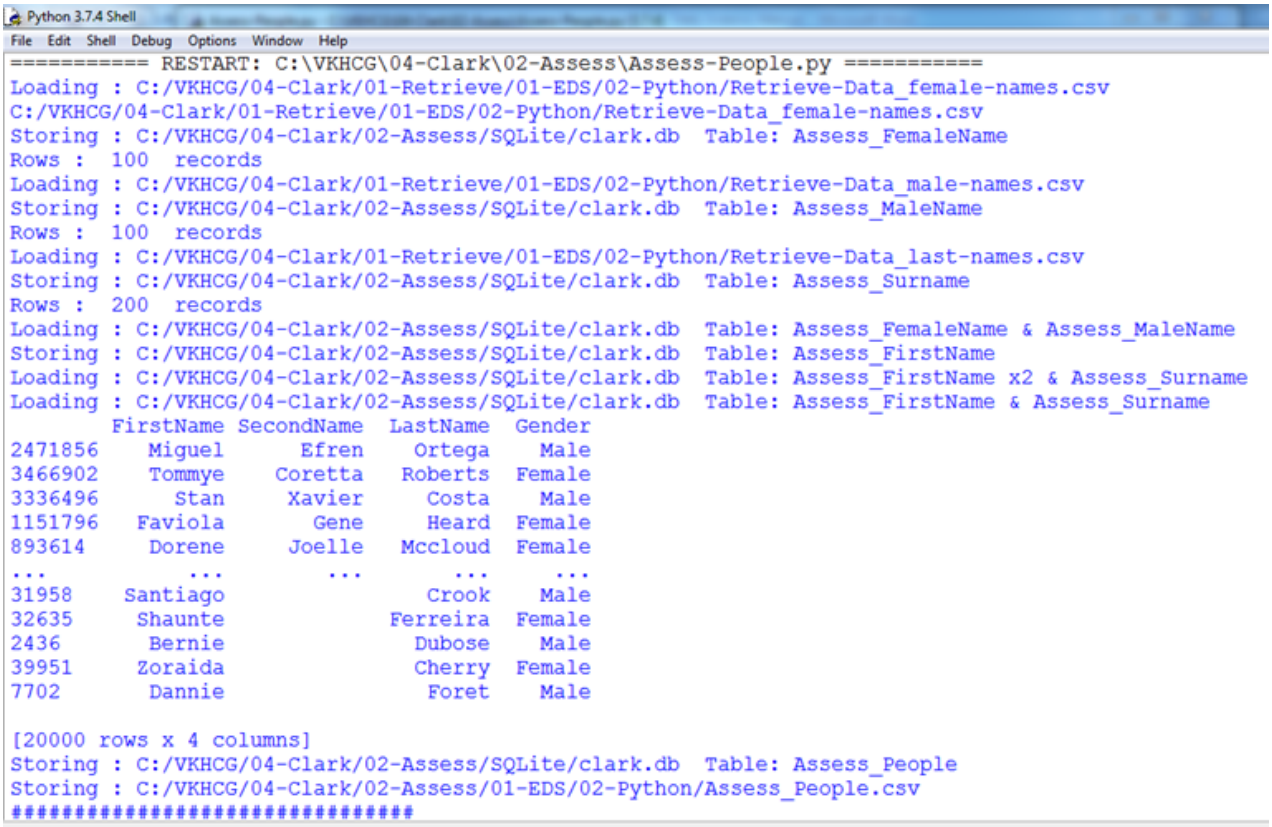
```
import sys import os
import sqlite3 as sq import pandas as pd
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
Company='04-Clark'
sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve-Data_female-names.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve-Data_male-names.csv'
sInputFileName3='01-Retrieve/01-EDS/02-Python/Retrieve-Data_last-names.csv'
sOutputFileName1='Assess-Staff.csv'
sOutputFileName2='Assess-Customers.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/clark.db' conn = sq.connect(sDatabaseName)
#####################################################################
### Import Female Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('##################################')
print('Loading :',sFileName) print('##################################')
print(sFileName)
FemaleRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
FemaleRawData.rename(columns={'NameValues' : 'FirstName'},inplace=True)
FemaleRawData.drop_duplicates(subset=None, keep='first', inplace=True)
FemaleData=FemaleRawData.sample(100)
print('##################################')
#####################################################################
print('#################')
sTable='Assess_FemaleName'
print('Storing :',sDatabaseName,' Table:',sTable)
```

```
FemaleData.to_sql(sTable, conn, if_exists="replace") print('################')
print('##################################')
print('Rows : ',FemaleData.shape[0], ' records') print('##################################')
#####################################################################
### Import Male Data
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('##################################')
print('Loading :',sFileName) print('##################################')
MaleRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
MaleRawData.rename(columns={'NameValues' : 'FirstName'},inplace=True)
MaleRawData.drop_duplicates(subset=None, keep='first', inplace=True)
MaleData=MaleRawData.sample(100)
print('##################################')
sTable='Assess_MaleName'
print('Storing :',sDatabaseName,' Table:',sTable) MaleData.to_sql(sTable, conn,
if_exists="replace") print('################')
#####################################################################
print('##################################')
print('Rows : ',MaleData.shape[0], ' records') print('##################################')
#####################################################################
### Import Surname Data
sFileName=Base + '/' + Company + '/' + sInputFileName3
print('##################################')
print('Loading :',sFileName) print('##################################')
SurnameRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
SurnameRawData.rename(columns={'NameValues' : 'LastName'},inplace=True)
SurnameRawData.drop_duplicates(subset=None, keep='first', inplace=True)
SurnameData=SurnameRawData.sample(200)
print('################')
sTable='Assess_Surname'
print('Storing :',sDatabaseName,' Table:',sTable) SurnameData.to_sql(sTable, conn,
if_exists="replace") print('################') print('##################################')
print('Rows : ',SurnameData.shape[0], ' records') print('##################################')
print('################')
sTable='Assess_FemaleName & Assess_MaleName' print('Loading :',sDatabaseName,'
Table:',sTable) sSQL="select distinct"
sSQL=sSQL+ " A.FirstName," sSQL=sSQL+ " 'Female' as Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FemaleName as A" sSQL=sSQL+ " UNION"
```

```
sSQL=sSQL+ " select distinct" sSQL=sSQL+ " A.FirstName," sSQL=sSQL+ " 'Male' as
Gender" sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_MaleName as A;" FirstNameData=pd.read_sql_query(sSQL, conn)
print('#################')
#########################################################################
#print('#################')
sTable='Assess_FirstName'
print('Storing :',sDatabaseName,' Table:',sTable) FirstNameData.to_sql(sTable, conn,
if_exists="replace") print('#################')
#########################################################################
#########################################################################
print('#################')
sTable='Assess_FirstName x2 & Assess_Surname' print('Loading :',sDatabaseName,'
Table:',sTable) sSQL="select distinct"
sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " B.FirstName AS SecondName," sSQL=sSQL+ " C.LastName,"
sSQL=sSQL+ " A.Gender" sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FirstName as A" sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_FirstName as B" sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Surname as C" sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " A.Gender = B.Gender" sSQL=sSQL+ " AND"
sSQL=sSQL+ " A.FirstName <> B.FirstName;" PeopleRawData=pd.read_sql_query(sSQL,
conn) People1Data=PeopleRawData.sample(10000)

sTable='Assess_FirstName & Assess_Surname' print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.FirstName," sSQL=sSQL+ " " AS SecondName," sSQL=sSQL+ "
B.LastName," sSQL=sSQL+ " A.Gender" sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FirstName as A"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Surname as B;" PeopleRawData=pd.read_sql_query(sSQL, conn)
People2Data=PeopleRawData.sample(10000) PeopleData=People1Data.append(People2Data)
print(PeopleData)
print('#################')
#########################################################################
#print('#################')
sTable='Assess_People'
print('Storing :',sDatabaseName,' Table:',sTable) PeopleData.to_sql(sTable, conn,
if_exists="replace") print('#################')
#########################################################################
```

```
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
########################################################################
sOutputFileName = sTable+'.csv' sFileName=sFileDir + '/' + sOutputFileName
print('####################################')
print('Storing :', sFileName) print('####################################')
PeopleData.to_csv(sFileName, index = False) print('####################################')
########################################################################
print('### Done!! ################################################')
```

Output:



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

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

Practical No 5: Build the time hub, links, and satellites.

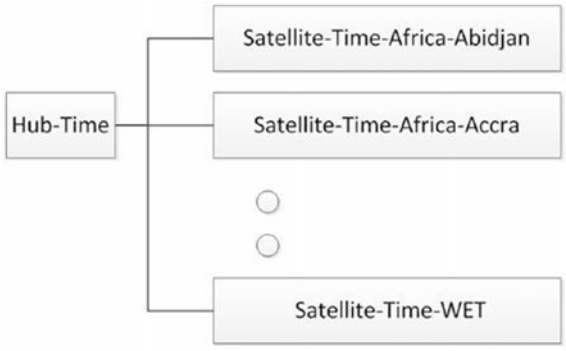
Code:

```
import sys import os
from datetime import datetime from datetime import timedelta
from pytz import timezone, all_timezones import pandas as pd
import sqlite3 as sq
from pandas.io import sql import uuid
pd.options.mode.chained_assignment = None
#####################################################################
if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG' print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
Company='01-Vermeulen' InputDir='00-RawData' InputFileName='VehicleData.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db' conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db' conn2 = sq.connect(sDatabaseName)
#####################################################################
base = datetime(2018,1,1,0,0,0)
numUnits=10*365*24
#####################################################################
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)] t=0
for i in date_list: now_utc=i.replace(tzinfo=timezone('UTC'))
sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S") print(sDateTime)
sDateTimeKey=sDateTime.replace(' ','-').replace(':','-') t+=1
IDNumber=str(uuid.uuid4()) TimeLine=[('ZoneBaseKey', ['UTC']),
('IDNumber', [IDNumber]), ('nDateTimeValue', [now_utc]), ('DateTimeValue', [sDateTime]),
('DateTimeKey', [sDateTimeKey])]
if t==1:
```

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

TimeZoneFrameIndex=TimeZoneFrame.set_index(['IDZoneNumber'],inplace=False)
sZone=zone.replace('/','-').replace(' ','')
#####################################################################
sTable = 'Process-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable) TimeZoneFrameIndex.to_sql(sTable, conn1,
if_exists="replace")
```

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

Output:



Golden Nominal

Code:

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

start_date = datetime(1900,1,1,0,0,0) start_date_utc=start_date.replace(tzinfo=timezone('UTC'))
HoursBirth=100*365*24 RawData['BirthDateUTC']=RawData.apply(lambda row:
```

```
(start_date_utc + timedelta(hours=randint(0, HoursBirth)))
,axis=1) zonemax=len(all_timezones)-1
RawData['TimeZone']=RawData.apply(lambda row: (all_timezones[randint(0, zonemax)])
,axis=1) RawData['BirthDateISO']=RawData.apply(lambda row:
row["BirthDateUTC"].astimezone(timezone(row['TimeZone']))
,axis=1) RawData['BirthDateKey']=RawData.apply(lambda row:
row["BirthDateUTC"].strftime("%Y-%m-%d  %H:%M:%S")
,axis=1) RawData['BirthDate']=RawData.apply(lambda row:
row["BirthDateISO"].strftime("%Y-%m-%d  %H:%M:%S")
,axis=1) RawData['PersonID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1) ######################################################################
Data=RawData.copy() Data.drop('BirthDateUTC', axis=1,inplace=True)
Data.drop('BirthDateISO', axis=1,inplace=True) indexed_data = Data.set_index(['PersonID'])

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

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

Output:

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

Load the person into the data vault

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

Vehicles

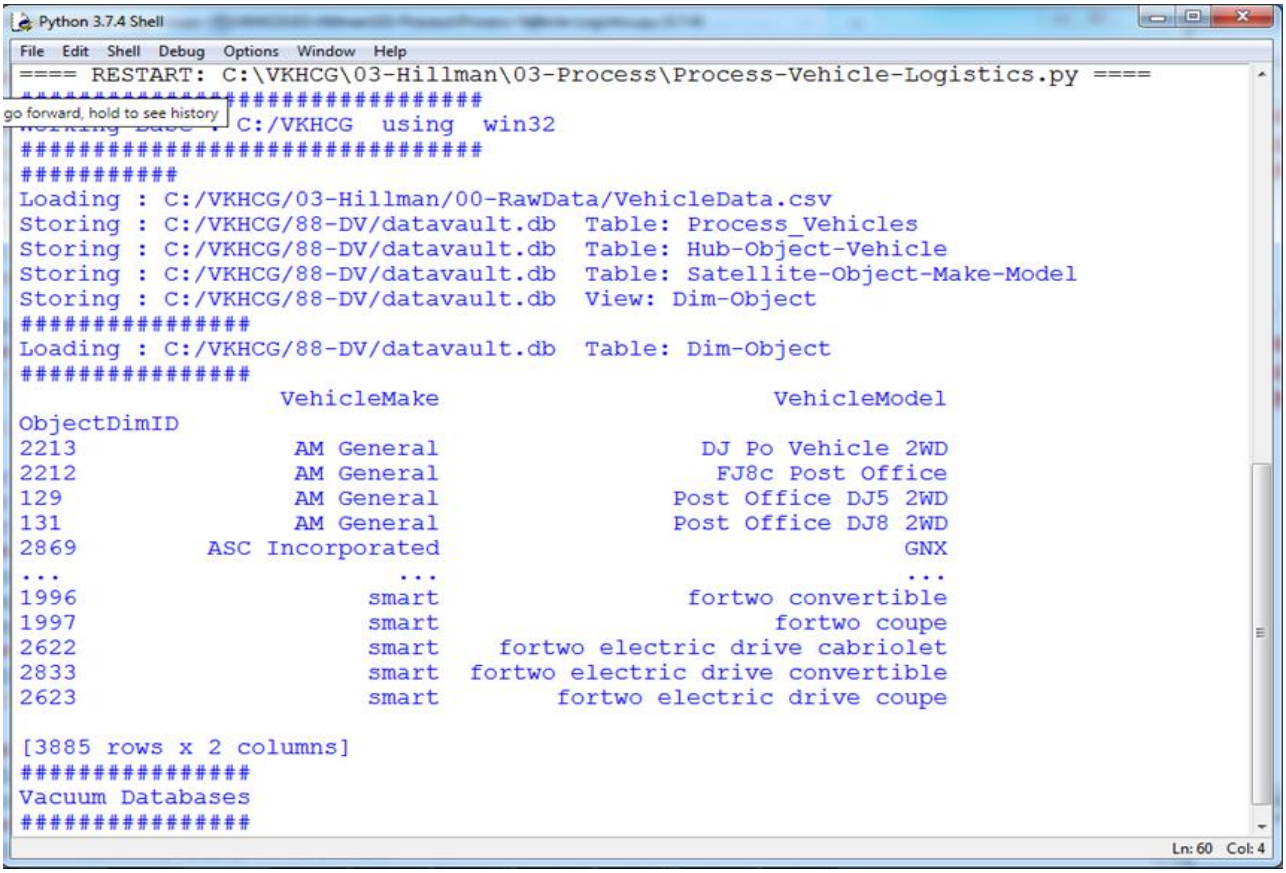
Code:

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

Output:



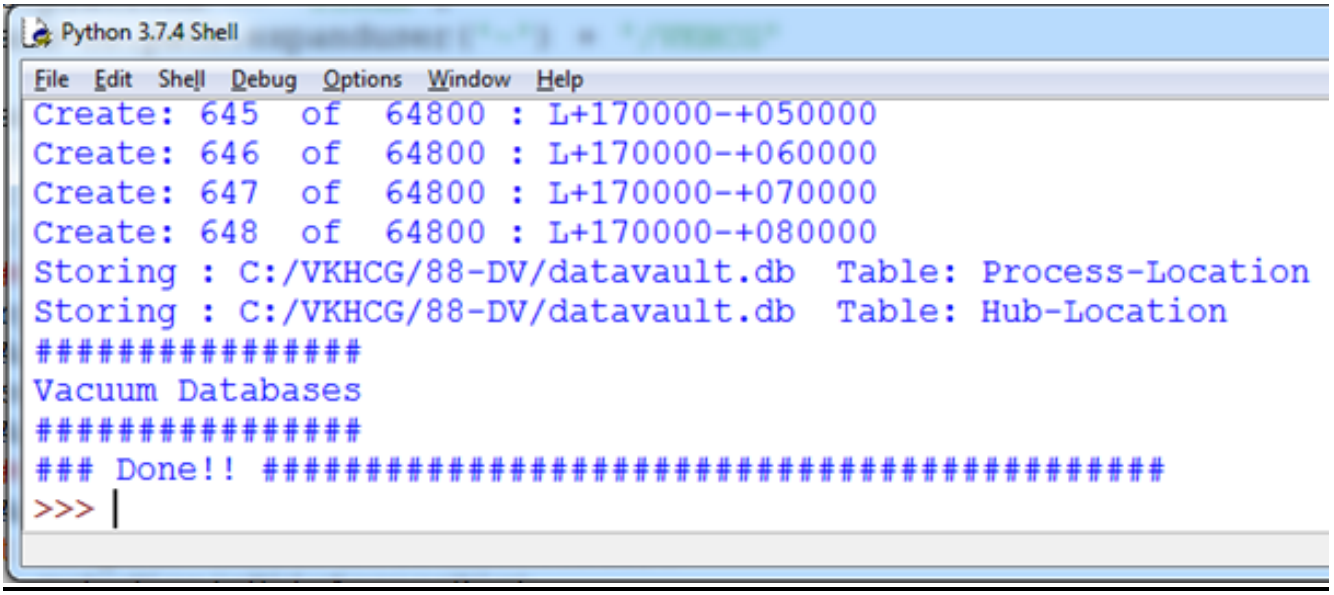
Human Environment Interaction

Code:

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

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

Output:



Forecasting

Code:

```
import sys import os
import sqlite3 as sq import quandl import pandas as pd
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
Company='04-Clark'
sInputFileName='00-RawData/VKHCG_Shares.csv' sOutputFileName='Shares.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sFileDir1=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python' if not
os.path.exists(sFileDir1):
os.makedirs(sFileDir1)
#####################################################################
sFileDir2=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not
os.path.exists(sFileDir2):
os.makedirs(sFileDir2)
#####################################################################
sFileDir3=Base + '/' + Company + '/03-Process/01-EDS/02-Python' if not
os.path.exists(sFileDir3):
os.makedirs(sFileDir3)
#####################################################################
sDatabaseName=sDataBaseDir + '/clark.db' conn = sq.connect(sDatabaseName)
#####################################################################
### Import Share Names Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName) print('##################################')
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Rows  :',RawData.shape[0]) print('Columns:',RawData.shape[1])
print('#################')
```

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

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

Output:

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

```
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Microsoft.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Microsoft.csv
Share  : WIKI/UPS
Rows   : 4622
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db  Table: WIKI_UPS
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_UPS.csv
Storing :  C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_UPS.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_UPS.csv
Share  : WIKI/AMZN
Rows   : 5248
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db  Table: WIKI_Amazon
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Amazon.csv
Storing :  C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Amazon.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Amazon.csv
Share  : LOCALBTC/USD
Rows   : 1863
Columns: 6
```

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

Practical No 6: Transforming Data

Transform Superstep

Code:

```
import sys import os
from datetime import datetime from pytz import timezone import pandas as pd
import sqlite3 as sq import uuid
pd.options.mode.chained_assignment = None
#######################################################################
Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#######################################################################
Company='01-Vermeulen' InputDir='00-RawData' InputFileName='VehicleData.csv'
#######################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite' if not
os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#######################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db' conn1 = sq.connect(sDatabaseName)
#######################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir): os.makedirs(sDataVaultDir)
#######################################################################
sDatabaseName=sDataVaultDir + '/datavault.db' conn2 = sq.connect(sDatabaseName)
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir): os.makedirs(sDataWarehouseDir)
#######################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn3 = sq.connect(sDatabaseName)
#######################################################################
print('\n#################################')
print('Time Category') print('UTC Time')
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
print(BirthDateZoneUTCStr)
```

```
print('#################################')
print('Birth Date in Reykjavik :') BirthZone = 'Atlantic/Reykjavik'
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S") print(BirthDateStr)
print('#################################')
#####################################################################
IDZoneNumber=str(uuid.uuid4()) sDateTimeKey=BirthDateZoneStr.replace(' ','-').replace(':','-')
TimeLine=[('ZoneBaseKey', ['UTC']),
('IDNumber', [IDZoneNumber]), ('DateTimeKey', [sDateTimeKey]), ('UTCDateTimeValue',
[BirthDateZoneUTC]), ('Zone', [BirthZone]),
('DateTimeValue', [BirthDateStr])] TimeFrame = pd.DataFrame.from_items(TimeLine)
#####################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Hub-Time-Gunnarsson' print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace") sTable = 'Dim-Time-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']]
TimeSatelliteIndex=TimeSatellite.set_index(['IDNumber'],inplace=False)
#####################################################################
BirthZoneFix=BirthZone.replace(' ','-').replace('/','-') sTable = 'Satellite-Time-' + BirthZoneFix +
'-Gunnarsson' print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeSatelliteIndex.to_sql(sTable, conn2, if_exists="replace") sTable = 'Dim-Time-' +
BirthZoneFix + '-Gunnarsson' TimeSatelliteIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
print('\n#################################')
print('Person Category') FirstName = 'Guðmundur' LastName = 'Gunnarsson'
print('Name:',FirstName,LastName) print('Birth Date:',BirthDateLocal) print('Birth
Zone:',BirthZone)
print('UTC Birth Date:',BirthDateZoneStr) print('#################################')
#####################################################################
IDPersonNumber=str(uuid.uuid4()) PersonLine=[('IDNumber', [IDPersonNumber]),
('FirstName', [FirstName]),
```

```
('LastName', [LastName]),
('Zone', ['UTC']),
('DateTimeValue', [BirthDateZoneStr])] PersonFrame = pd.DataFrame.from_items(PersonLine)
#####################################################################
TimeHub=PersonFrame TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Hub-Person-Gunnarsson' print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace") sTable = 'Dim-Person-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
```

Output:

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

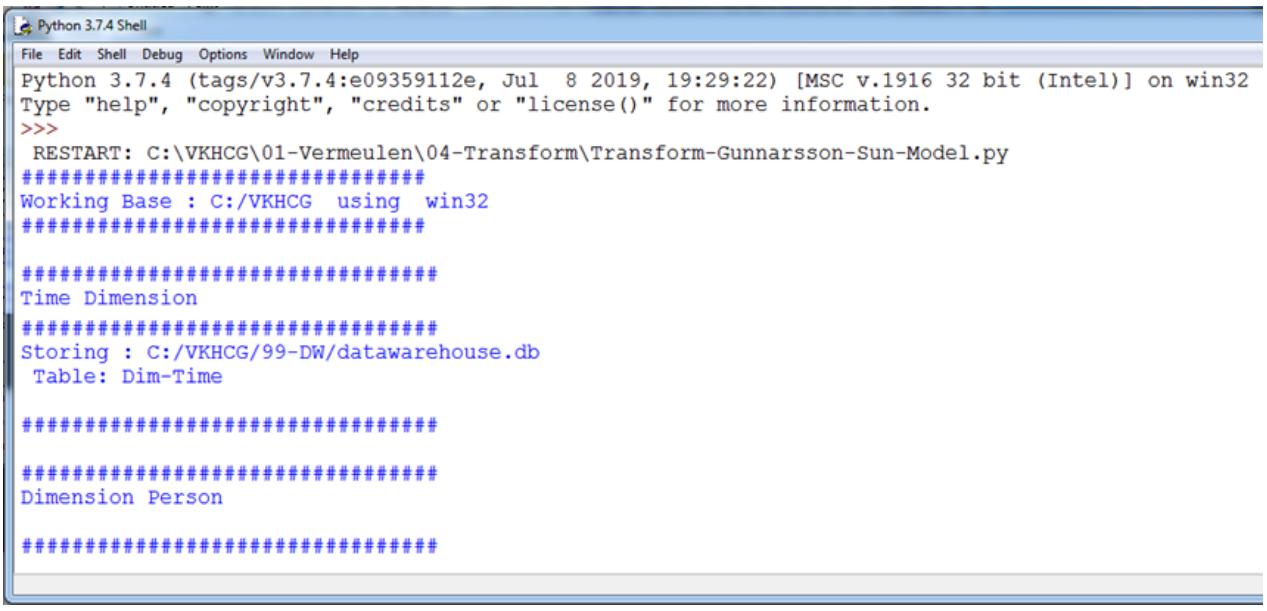
```
>>>
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson_is_Born.py
Working Base : C:/VKHCG  using  win32
Time Category
UTC Time
1960-12-20 10:15:00 (UTC) (+0000)
###################################
Birth Date in Reykjavik :
1960-12-20 09:15:00 (-01) (-0100)
###################################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

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

```
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)
```

Output:



Building a Data Warehouse

Code:

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

```
BirthZone = ('Atlantic/Reykjavik','Europe/London','UCT') for j in range(len(BirthZone)):
t+=1
print(t,mt*3)
BirthDateUTC = datetime.strptime(DateData['DateTimeValue'][i],"%Y-%m-%d %H:%M:%S")
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone[j]))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
#####################################################################
IDTimeNumber=str(uuid.uuid4()) TimeLine=[('TimeID', [str(IDTimeNumber)]),
('UTCDate', [str(BirthDateZoneStr)]), ('LocalTime', [str(BirthDateLocal)]), ('TimeZone',
[str(BirthZone)])]
if t==1:
TimeFrame = pd.DataFrame.from_items(TimeLine) else:
TimeRow = pd.DataFrame.from_items(TimeLine) TimeFrame=TimeFrame.append(TimeRow)
#####################################################################
DimTime=TimeFrame DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
#####################################################################
sTable = 'Dim-Time' print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace") DimTimeIndex.to_sql(sTable, conn3,
if_exists="replace")
##################################################################### sSQL=" SELECT
" + \
" FirstName," + \
" SecondName," + \ " LastName," + \
" BirthDateKey " + \
" FROM [Hub-Person];" PersonDataRaw=pd.read_sql_query(sSQL, conn2)
PersonData=PersonDataRaw.head(1000)
print('\n###################################')
print('Dimension Person') print('\n###################################')
t=0 mt=DateData.shape[0] for i in range(mt):
t+=1
print(t,mt)
FirstName = str(PersonData["FirstName"]) SecondName = str(PersonData["SecondName"]) if
len(SecondName) > 0:
SecondName=""
```

```
LastName = str(PersonData["LastName"]) BirthDateKey = str(PersonData["BirthDateKey"])
#####################################################################
IDPersonNumber=str(uuid.uuid4()) PersonLine=[('PersonID', [str(IDPersonNumber)]),
('FirstName', [FirstName]), ('SecondName', [SecondName]), ('LastName', [LastName]), ('Zone',
[str('UTC')]),
('BirthDate', [BirthDateKey])]
if t==1:
PersonFrame = pd.DataFrame.from_items(PersonLine) else:
PersonRow = pd.DataFrame.from_items(PersonLine) PersonFrame =
PersonFrame.append(PersonRow)
#####################################################################
DimPerson=PersonFrame print(DimPerson)
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####################################################################
sTable = 'Dim-Person' print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace") DimPersonIndex.to_sql(sTable,
conn3, if_exists="replace")
```

Output:

You have successfully performed data vault to data warehouse transformation

Simple Linear Regression

Code:

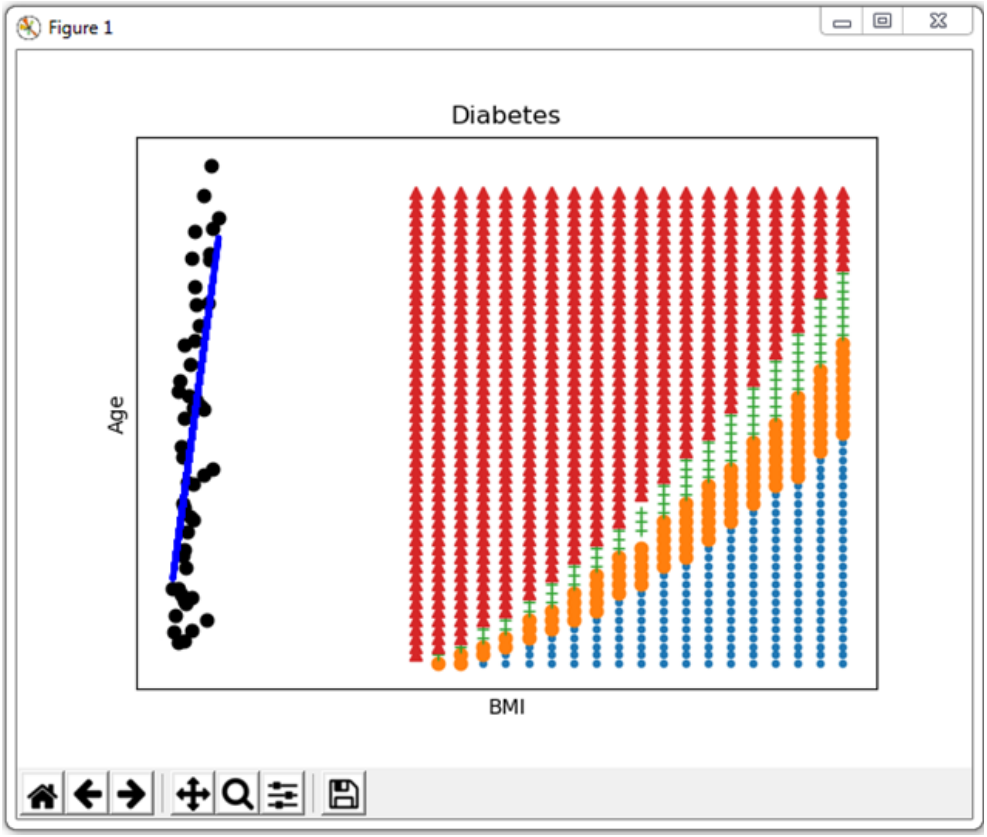
```
import sys import os
import pandas as pd import sqlite3 as sq
import matplotlib.pyplot as plt import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base,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

```
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir): os.makedirs(sDataVaultDir)
sDatabaseName=sDataVaultDir + '/datavault.db' conn2 = sq.connect(sDatabaseName)
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir): os.makedirs(sDataWarehouseDir)
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn3 = sq.connect(sDatabaseName)
t=0
tMax=((300-100)/10)*((300-30)/5)
for heightSelect in range(100,300,10): for weightSelect in range(30,300,5):
height = round(heightSelect/100,3) weight = int(weightSelect)
bmi = weight/(height*height) if bmi <= 18.5:
BMI_Result=1
elif bmi > 18.5 and bmi < 25:
BMI_Result=2
elif bmi > 25 and bmi < 30:
BMI_Result=3 elif bmi > 30:
BMI_Result=4 else:
BMI_Result=0 PersonLine=[('PersonID', [str(t)]),
('Height', [height]),
('Weight', [weight]),
('bmi', [bmi]),
('Indicator', [BMI_Result])]
t+=1
print('Row:',t,'of',tMax) if t==1:
PersonFrame = pd.DataFrame.from_items(PersonLine) else:
PersonRow = pd.DataFrame.from_items(PersonLine) PersonFrame =
PersonFrame.append(PersonRow)
DimPerson=PersonFrame DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
sTable = 'Transform-BMI' print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
sTable = 'Person-Satellite-BMI' print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI' print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
```

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

Output:

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

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

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

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

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

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

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



Practical No 7: Organizing Data

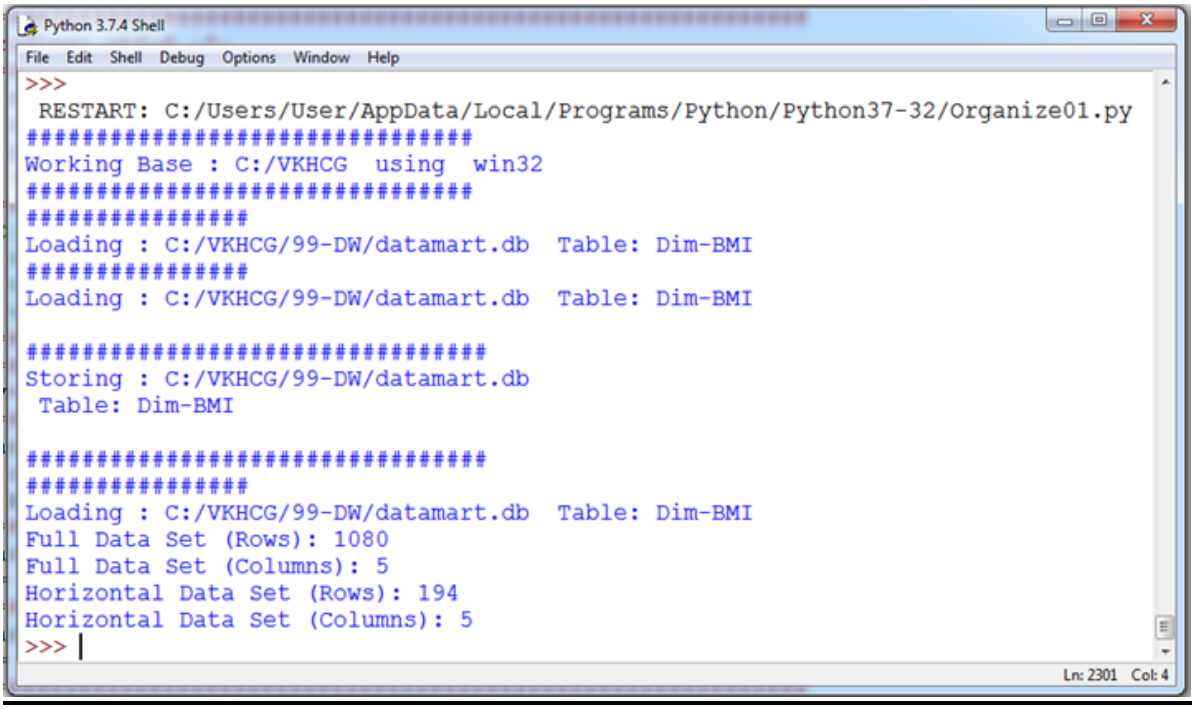
Horizontal Style

Code:

```
import sys import os
import pandas as pd import sqlite3 as sq
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir): os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db' conn2 = sq.connect(sDatabaseName)
#####################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1) print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT PersonID,\
Height,\ Weight,\ bmi,\ Indicator\
FROM [Dim-BMI]\ WHERE \
Height > 1.5 \ and Indicator = 1\ ORDER BY \
Height,\ Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####################################################################
DimPerson=PersonFrame1 DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####################################################################
sTable = 'Dim-BMI' print('\n##################################')
```

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

Output:



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

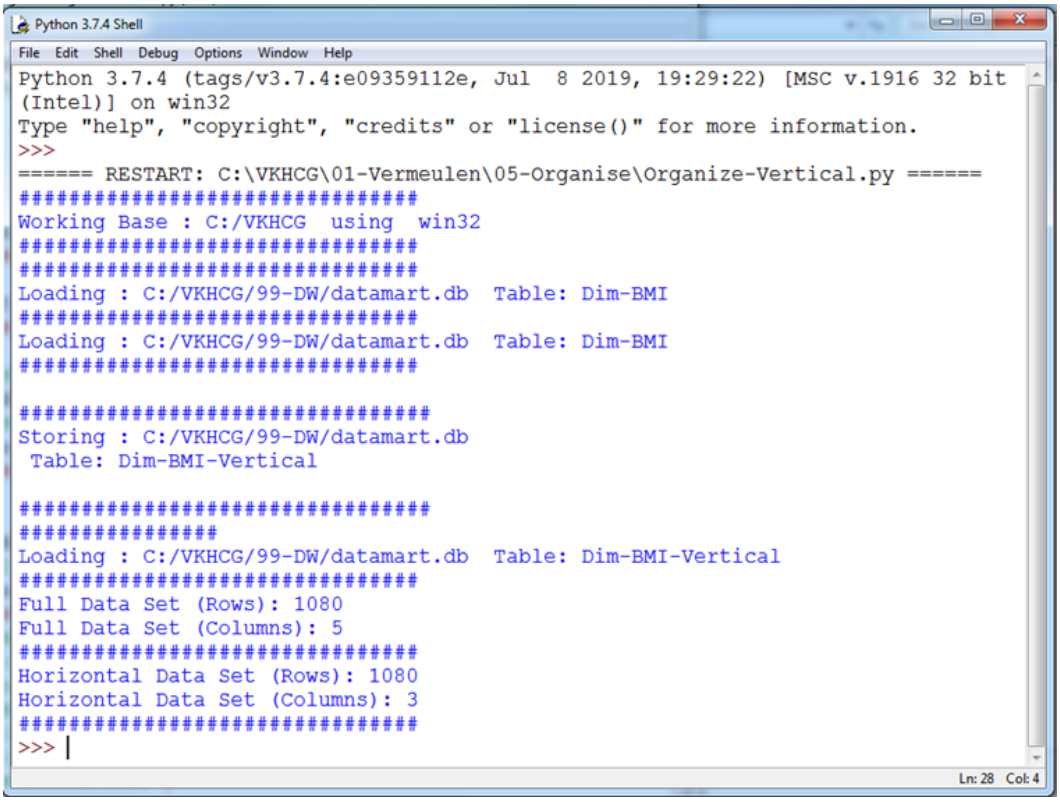
Vertical Style

Code:

```
import sys import os
import pandas as pd import sqlite3 as sq
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir): os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db' conn2 = sq.connect(sDatabaseName)
#####################################################################
print('##################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####################################################################
print('##################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) print('##################################')
sSQL="SELECT \
Height,\ Weight,\ Indicator\
FROM [Dim-BMI];"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####################################################################
DimPerson=PersonFrame1 DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
sTable = 'Dim-BMI-Vertical' print('\n##################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n##################################')
```

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

Output:



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

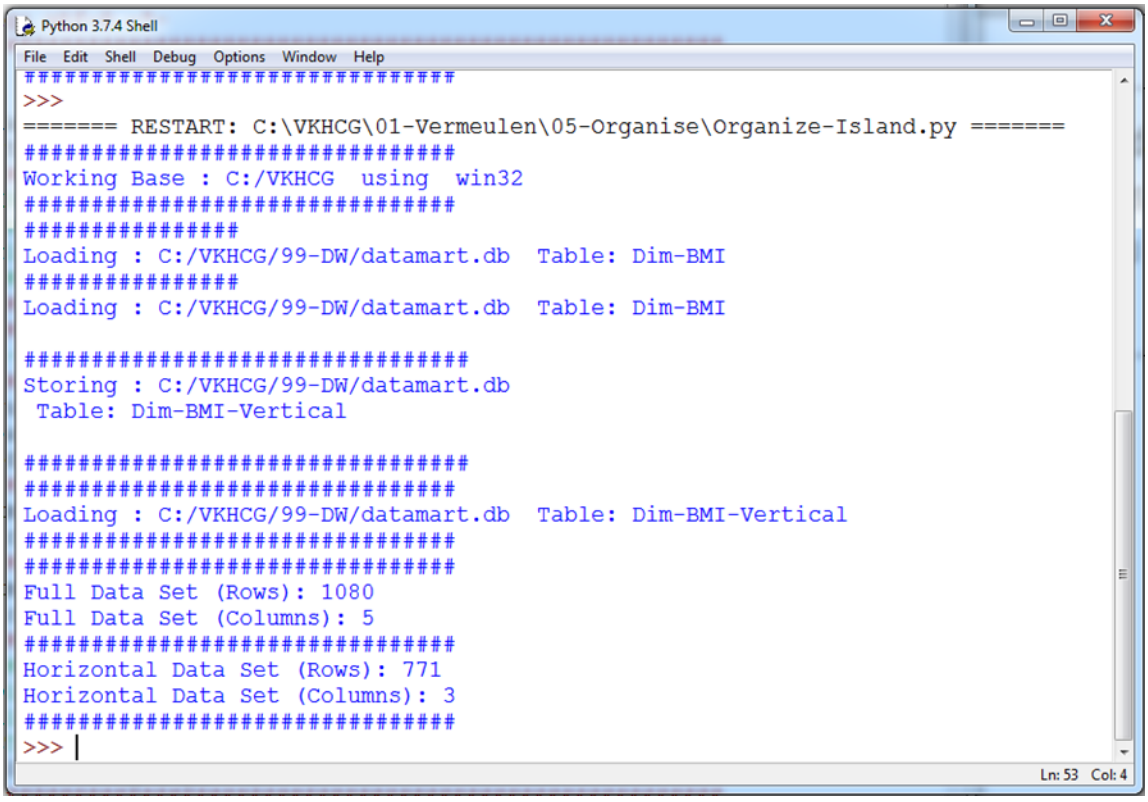
Island Style

Code:

```
import sys import os
import pandas as pd import sqlite3 as sq
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
Company='01-Vermeulen'
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir): os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db' conn2 = sq.connect(sDatabaseName)
#####################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
Height,\ Weight,\ Indicator\
FROM [Dim-BMI]\
WHERE Indicator > 2\ ORDER BY  \
Height,\ Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####################################################################
DimPerson=PersonFrame1 DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####################################################################
sTable = 'Dim-BMI-Vertical' print('\n##################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n##################################')
```

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

Output:



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

Secure Vault Style

Code:

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

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

Output:

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

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

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

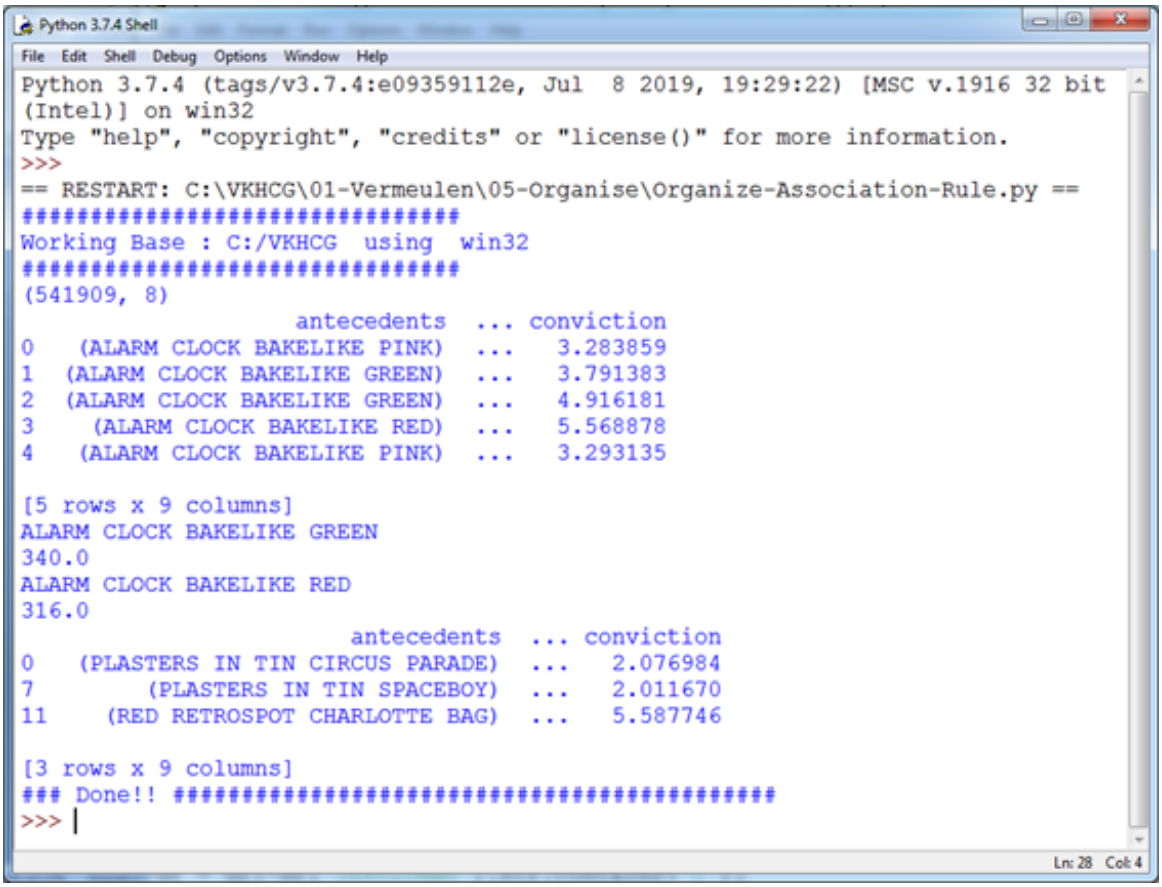
Assoiation Rule Mining

Code:

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

```
basket_sets2 = basket2.applymap(encode_units) basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True) rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)
print(rules2[ (rules2['lift'] >= 4) & (rules2['confidence'] >= 0.5)])
print('### Done!! ################################################')
```

Output:



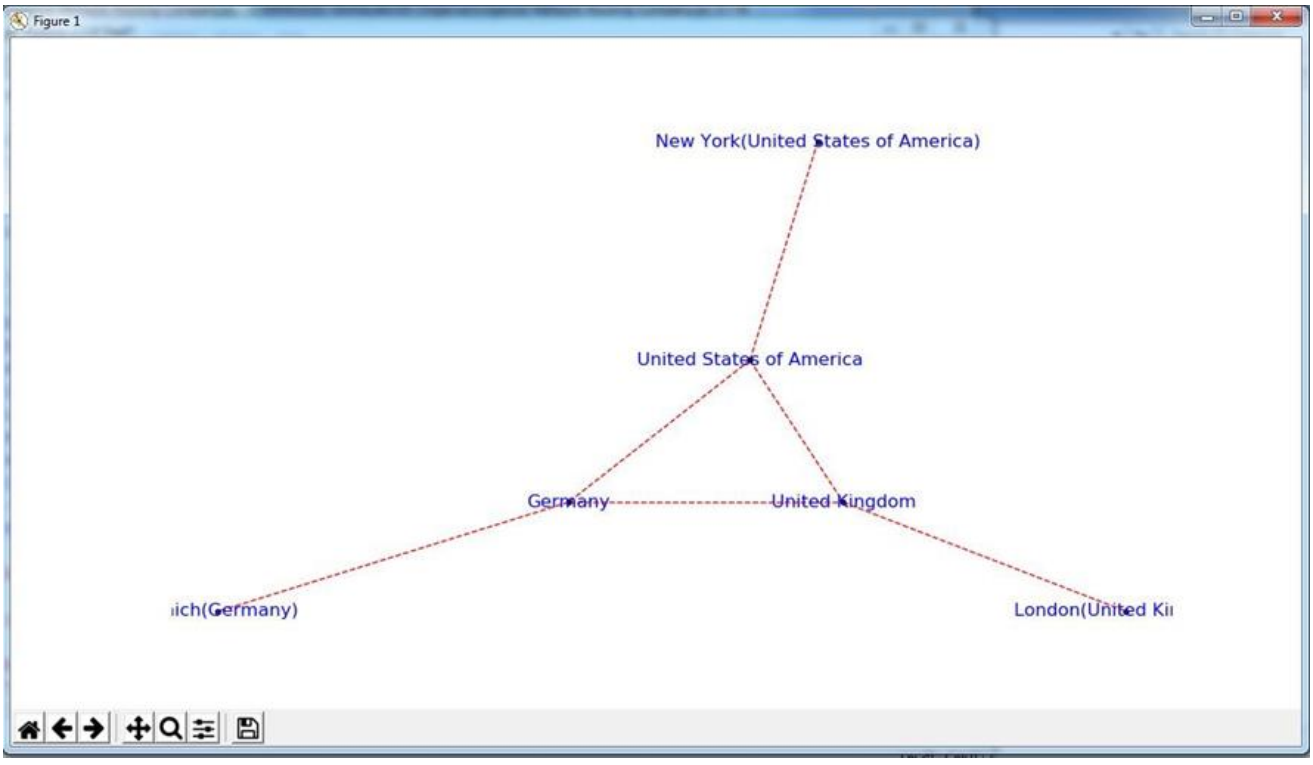
Create a Network Diagran

Code:

```
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] + '('+ 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('####################################') CompanyData['Company_Country_Name'][i] + ')' if
Node0 != Node1:
```

Output:



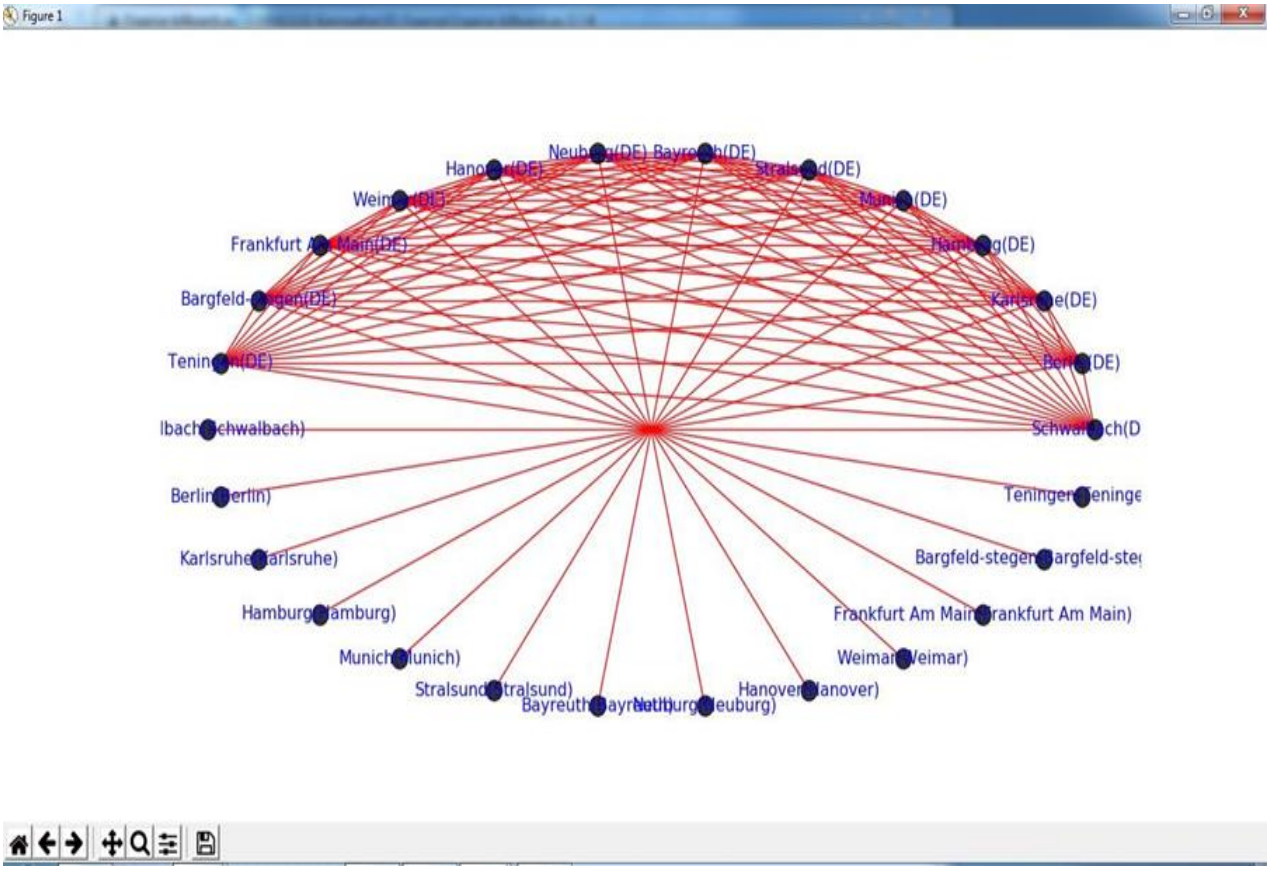
Picking Content for Billboards

Code:

```
import sys import os
import pandas as pd import networkx as nx
import matplotlib.pyplot as plt import numpy as np
pd.options.mode.chained_assignment = None 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'+ sOutputFileName1
print('####################################')
```

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

Output:

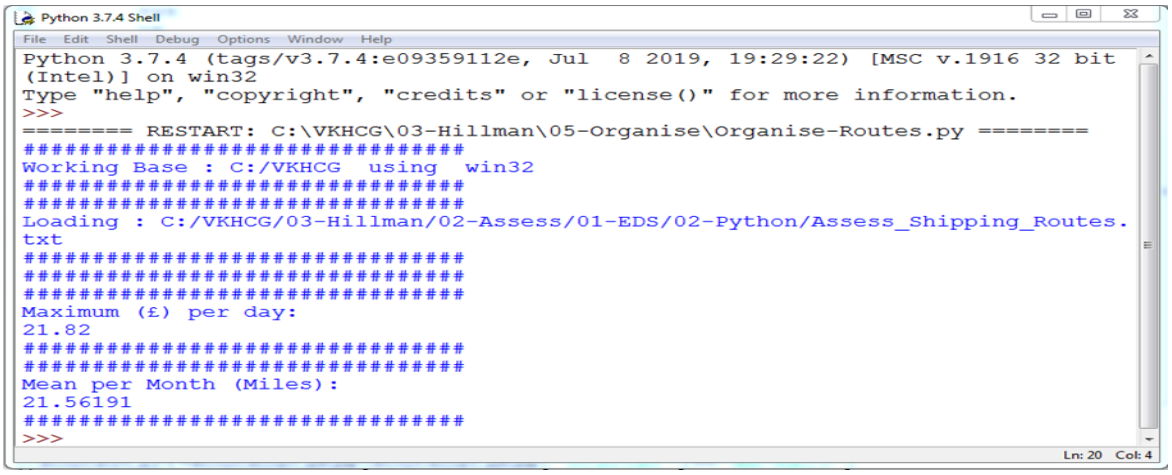


Create a delivery route

Code:

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

Output:


Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\VKHCG\03-Hillman\05-Organise\Organise-Routes.py ========
################################
Working Base : C:/VKHCG using win32
################################
################################
Loading : C:/VKHCG/03-Hillman/02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.txt
################################
################################
################################
Maximum (£) per day:
21.82
################################
################################
Mean per Month (Miles):
21.56191
################################
>>>
Ln: 20 Col: 4

Simple Forex Trading Planner

Code:

```
import sys import os
import pandas as pd import sqlite3 as sq import re
Base='C:/VKHCG' print('####################################')
print('Working Base :',Base, ' using ', sys.platform)
print('####################################')
sInputFileName='03-Process/01-EDS/02-Python/Process_ExchangeRates.csv'
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Forex.csv'
Company='04-Clark'
sDatabaseName=Base + '/' + Company + '/05-Organise/SQLite/clark.db' conn =
sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
sFileName=Base + '/' + Company + '/' + sInputFileName
print('####################################')
print('Loading :',sFileName) print('####################################')
ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('####################################')
ForexDataRaw.index.names = ['RowID'] sTable='Forex_All'
print('Storing :',sDatabaseName,' Table:',sTable) ForexDataRaw.to_sql(sTable, conn,
if_exists="replace")
sSQL="SELECT 1 as Bag\
, CAST(min(Date) AS VARCHAR(10)) as Date \
,CAST(1000000.0000000 as NUMERIC(12,4)) as Money \
,'USD' as Currency \ FROM Forex_All \
;"
sSQL=re.sub("\s\s+", " ", sSQL) nMoney=pd.read_sql_query(sSQL, conn)
nMoney.index.names = ['RowID'] sTable='MoneyData'
print('Storing :',sDatabaseName,' Table:',sTable) nMoney.to_sql(sTable, conn,
if_exists="replace")
sTable='TransactionData'
print('Storing :',sDatabaseName,' Table:',sTable) nMoney.to_sql(sTable, conn,
if_exists="replace")
ForexDay=pd.read_sql_query("SELECT Date FROM Forex_All GROUP BY Date;", conn)
t=0
for i in range(ForexDay.shape[0]): sDay1=ForexDay['Date'][i] sDay=str(sDay1)
sSQL='\
SELECT M.Bag as Bag, \
```

```
F.Date as Date, \
round(M.Money * F.Rate,6) AS Money, \ F.CodeIn AS PCurrency, \
F.CodeOut AS Currency \ FROM MoneyData AS M \ JOIN \
( \
SELECT \
CodeIn, CodeOut, Date, Rate \ FROM \
Forex_All \ WHERE\
CodeIn = "USD" AND CodeOut = "GBP" \ UNION \
SELECT \
CodeOut AS CodeIn, CodeIn AS CodeOut, Date, (1/Rate) AS Rate \ FROM \
Forex_All \ WHERE\
CodeIn = "USD" AND CodeOut = "GBP" \
) AS F \
ON \ M.Currency=F.CodeIn \ AND \
F.Date ='" +sDay + "';'
sSQL=re.sub("\s\s+", " ", sSQL)
ForexDayRate=pd.read_sql_query(sSQL, conn) for j in range(ForexDayRate.shape[0]):
sBag=str(ForexDayRate['Bag'][j]) nMoney=str(round(ForexDayRate['Money'][j],2))
sCodeIn=ForexDayRate['PCurrency'][j] sCodeOut=ForexDayRate['Currency'][j]
sSQL='UPDATE MoneyData SET Date= '" + sDay + "', '
sSQL= sSQL + ' Money = ' + nMoney + ', Currency='" + sCodeOut + "'" sSQL= sSQL + '
WHERE Bag=' + sBag + ' AND Currency='" + sCodeIn + "';'
sSQL=re.sub("\s\s+", " ", sSQL) cur = conn.cursor() cur.execute(sSQL) conn.commit()
t+=1
print('Trade :', t, sDay, sCodeOut, nMoney)
sSQL=' \
INSERT INTO TransactionData ( \
RowID, \
Bag, \ Date, \ Money, \ Currency \
) \
SELECT ' + str(t) + ' AS RowID, \
Bag, \ Date, \ Money, \ Currency \
FROM MoneyData \;'
sSQL=re.sub("\s\s+", " ", sSQL)
cur = conn.cursor() cur.execute(sSQL) conn.commit()
sSQL="SELECT RowID, Bag, Date, Money, Currency FROM TransactionData ORDER BY
RowID;"
sSQL=re.sub("\s\s+", " ", sSQL) TransactionData=pd.read_sql_query(sSQL, conn)
OutputFile=Base + '/' + Company + '/' + sOutputFileName TransactionData.to_csv(OutputFile,
index = False)
```

Output:

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

Practical No 8: Generating Data

Report Superstep

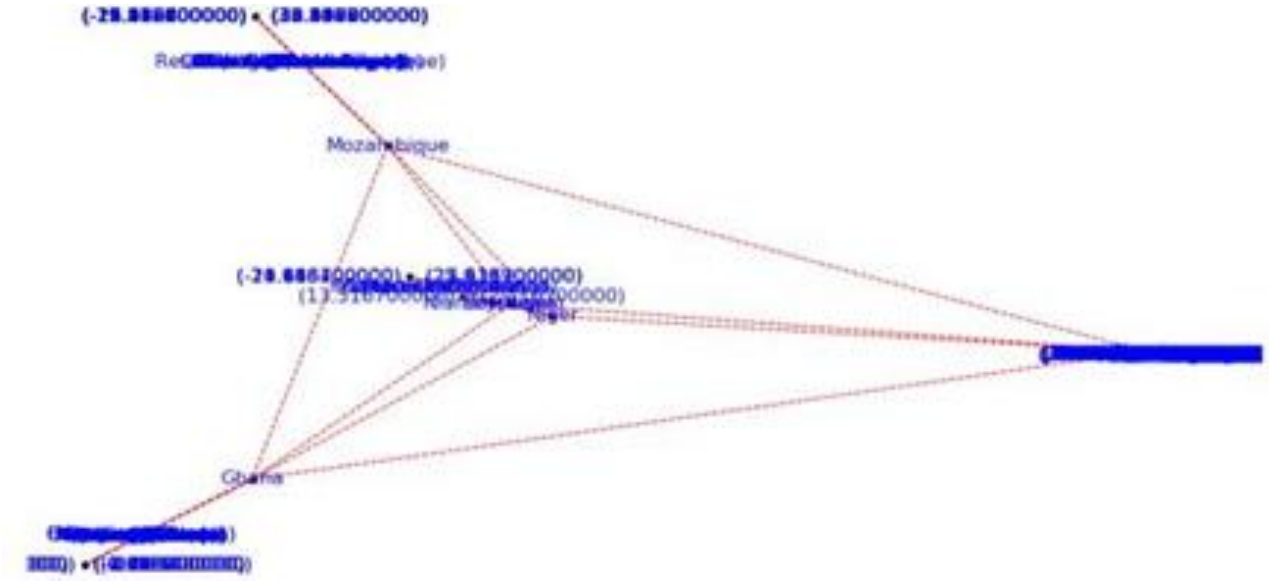
Vermeulen PLC

Code:

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

```
Node2='('+ "{:.9f}".format(CustomerData['Customer_Latitude'][i]) + ')\ ('+
"{:.9f}".format(CustomerData['Customer_Longitude'][i]) + ')'
if Node0 != Node1: G.add_edge(Node0,Node1)
if Node1 != Node2: G.add_edge(Node1,Node2)
print('Nodes:', G.number_of_nodes()) print('Edges:', G.number_of_edges())
sFileName=Base + '/' + Company + '/' + sOutputFileName1
print('###################################')
print('Storing :',sFileName) print('###################################')
nx.write_gml(G, sFileName)
sFileName=Base + '/' + Company + '/' + sOutputFileName2
print('###################################')
print('Storing Graph Image:',sFileName) print('###################################')
plt.figure(figsize=(25, 25)) pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600) plt.show()
print('###################################')
 print('### Done!! ####################')
print('###################################')
```

Output:



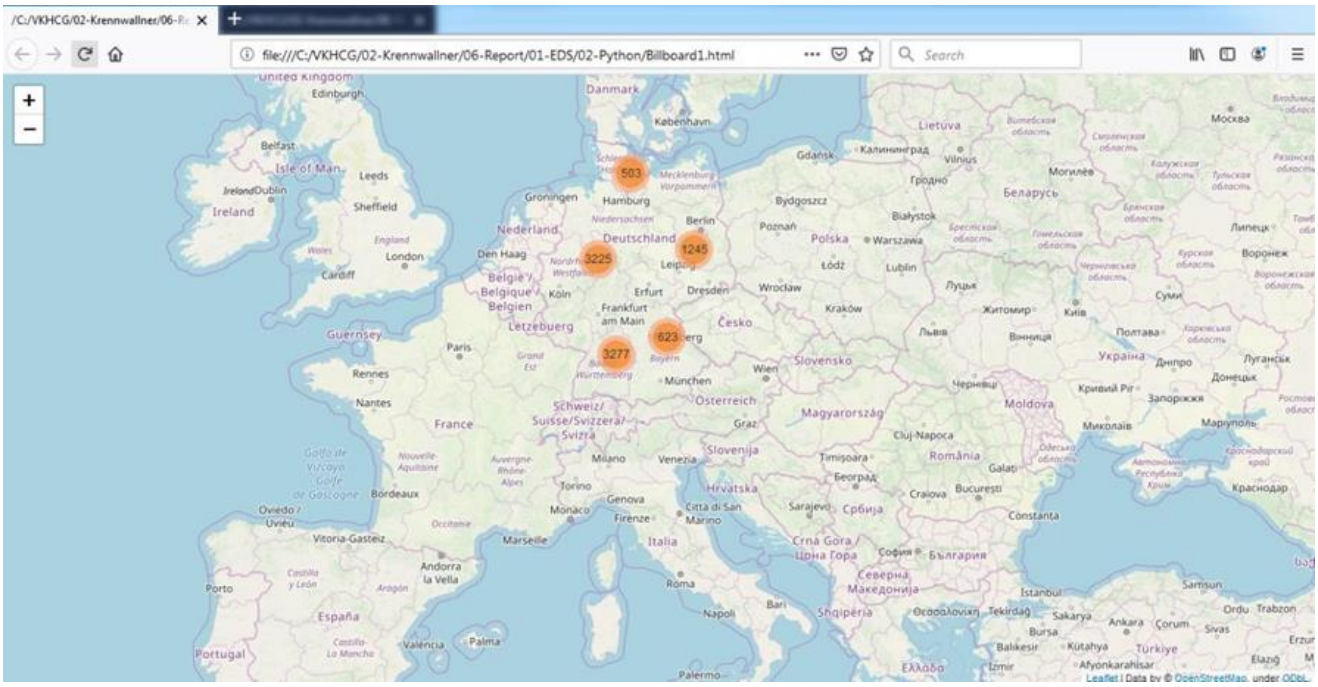
Krennwaller AG

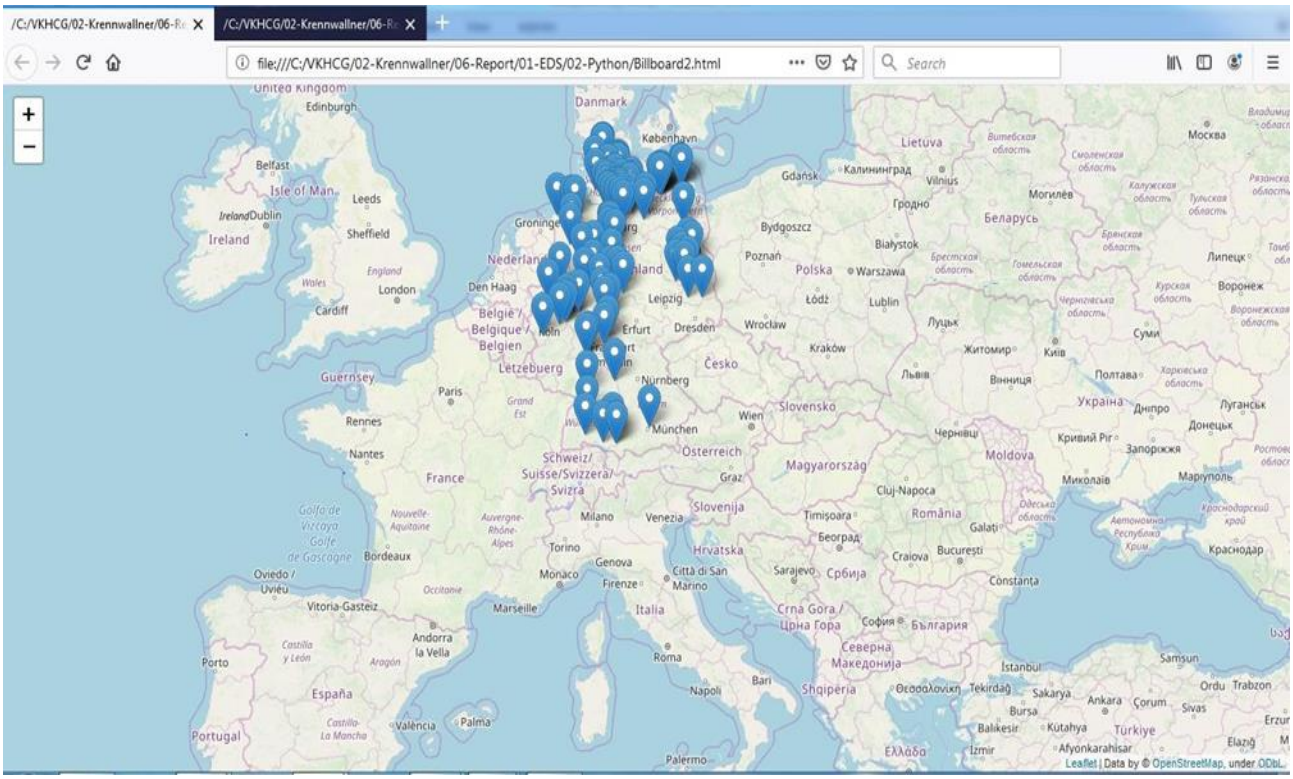
Code:

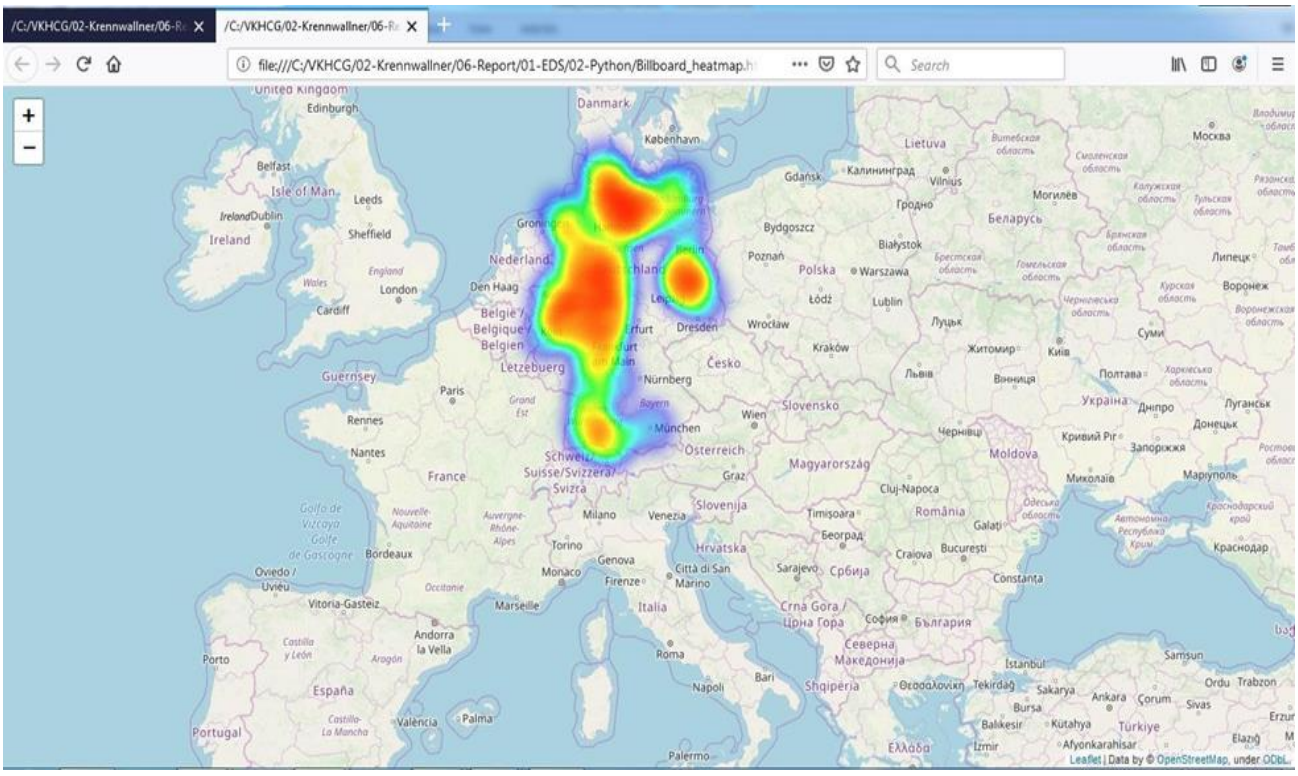
```
import sys import os
import pandas as pd
from folium.plugins import FastMarkerCluster, HeatMap from folium import Marker, Map
import webbrowser Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
sFileName=Base+'/02-Krennwallner/01-Retrieve/01-EDS/02-
Python/Retrieve_DE_Billboard_Locations.csv' df =
pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
df.fillna(value=0, inplace=True) print(df.shape)
t=0
for i in range(df.shape[0]): try:
sLongitude=df["Longitude"][i] sLongitude=float(sLongitude)
except Exception: sLongitude=float(0.0)
try:
sLatitude=df["Latitude"][i] sLatitude=float(sLatitude)
except Exception: sLatitude=float(0.0)
try:
sDescription=df["Place_Name"][i] + ' (' + df["Country"][i]+')'
except Exception: sDescription='VKHCG'
if sLongitude != 0.0 and sLatitude != 0.0: DataClusterList=list([sLatitude, sLongitude])
DataPointList=list([sLatitude, sLongitude, sDescription]) t+=1
if t==1:
DataCluster=[DataClusterList] DataPoint=[DataPointList]
else:
DataCluster.append(DataClusterList) DataPoint.append(DataPointList)
data=DataCluster pins=pd.DataFrame(DataPoint)
pins.columns = [ 'Latitude','Longitude','Description']
stops_map1 = Map(location=[48.1459806, 11.4985484], zoom_start=5) marker_cluster =
FastMarkerCluster(data).add_to(stops_map1)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard1.html'
stops_map1.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
stops_map2 = Map(location=[48.1459806, 11.4985484], zoom_start=5) for name, row in
pins.iloc[:100].iterrows():
```

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

Output:




file:///C:/VKHCG/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html
Search
Leaflet | Data by © OpenStreetMap, under ODbL.


file:///C:/VKHCG/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard_heatmap.h
Search
Leaflet | Data by © OpenStreetMap, under ODbL.

Hillman Ltd

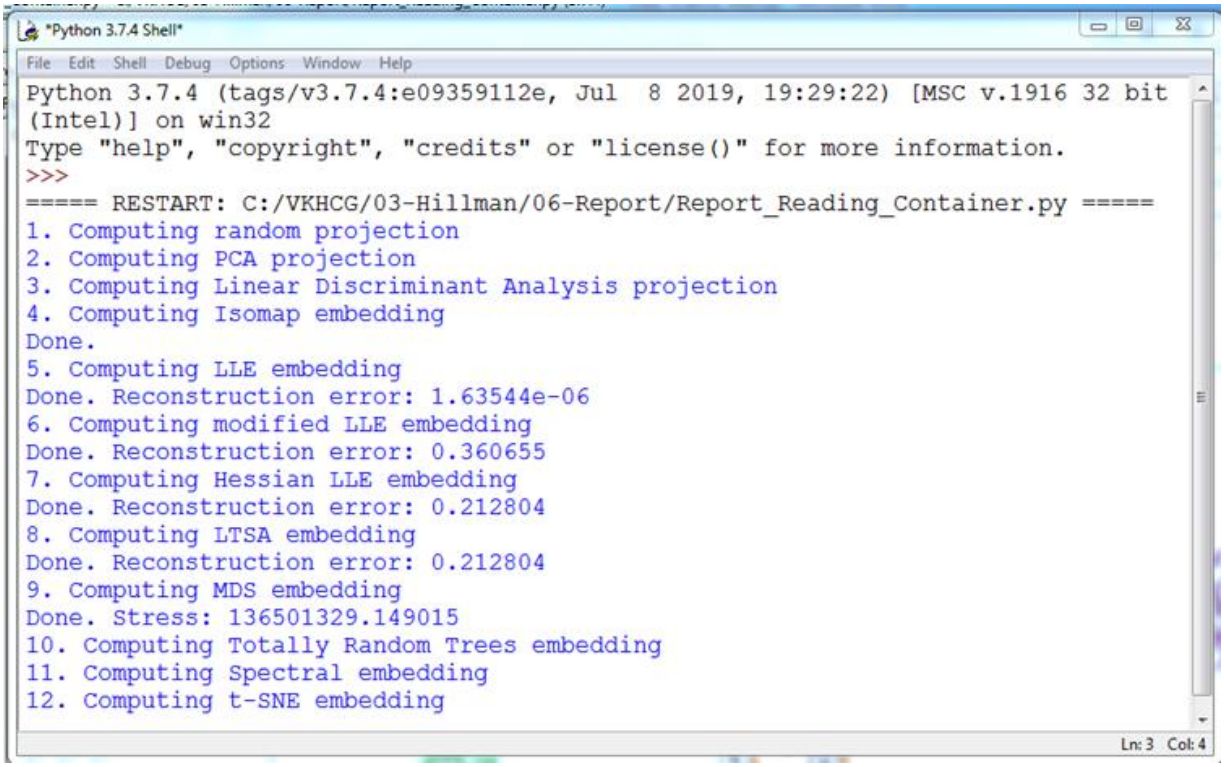
Code:

```
from time import time import numpy as np
import matplotlib.pyplot as plt from matplotlib import offsetbox
from sklearn import (manifold, datasets, decomposition, ensemble, discriminant_analysis,
random_projection) digits = datasets.load_digits(n_class=6)
X = digits.data y = digits.target
n_samples, n_features = X.shape n_neighbors = 30
def plot_embedding(X, title=None):
x_min, x_max = np.min(X, 0), np.max(X, 0) X = (X - x_min) / (x_max - x_min)
plt.figure(figsize=(10, 10))
ax = plt.subplot(111)
for i in range(X.shape[0]):
plt.text(X[i, 0], X[i, 1], str(digits.target[i]),
color=plt.cm.Set1(y[i] / 10.), fontdict={'weight': 'bold', 'size': 9})
if hasattr(offsetbox, 'AnnotationBbox'):
# only print thumbnails with matplotlib > 1.0 shown_images = np.array([[1., 1.]]) # just
something big for i in range(digits.data.shape[0]):
dist = np.sum((X[i] - shown_images) ** 2, 1) if np.min(dist) < 4e-3:
# don't show points that are too close continue
shown_images = np.r_[shown_images, [X[i]]]
imagebox = offsetbox.AnnotationBbox(offsetbox.OffsetImage(digits.images[i],
cmap=plt.cm.gray_r),X[i])
ax.add_artist(imagebox) plt.xticks([]), plt.yticks([]) if title is not None:
plt.title(title) n_img_per_row = 20
img = np.zeros((10 * n_img_per_row, 10 * n_img_per_row)) for i in range(n_img_per_row):
ix = 10 * i + 1
for j in range(n_img_per_row): iy = 10 * j + 1
img[ix:ix + 8, iy:iy + 8] = X[i * n_img_per_row + j].reshape((8, 8)) plt.figure(figsize=(10, 10))
plt.imshow(img, cmap=plt.cm.binary) plt.xticks([])
plt.yticks([])
plt.title('A selection from the 64-dimensional digits dataset') print("Computing random
projection")
rp = random_projection.SparseRandomProjection(n_components=2, random_state=42)
X_projected = rp.fit_transform(X)
plot_embedding(X_projected, "Random Projection of the digits") print("Computing PCA
projection")
t0 = time()
```

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

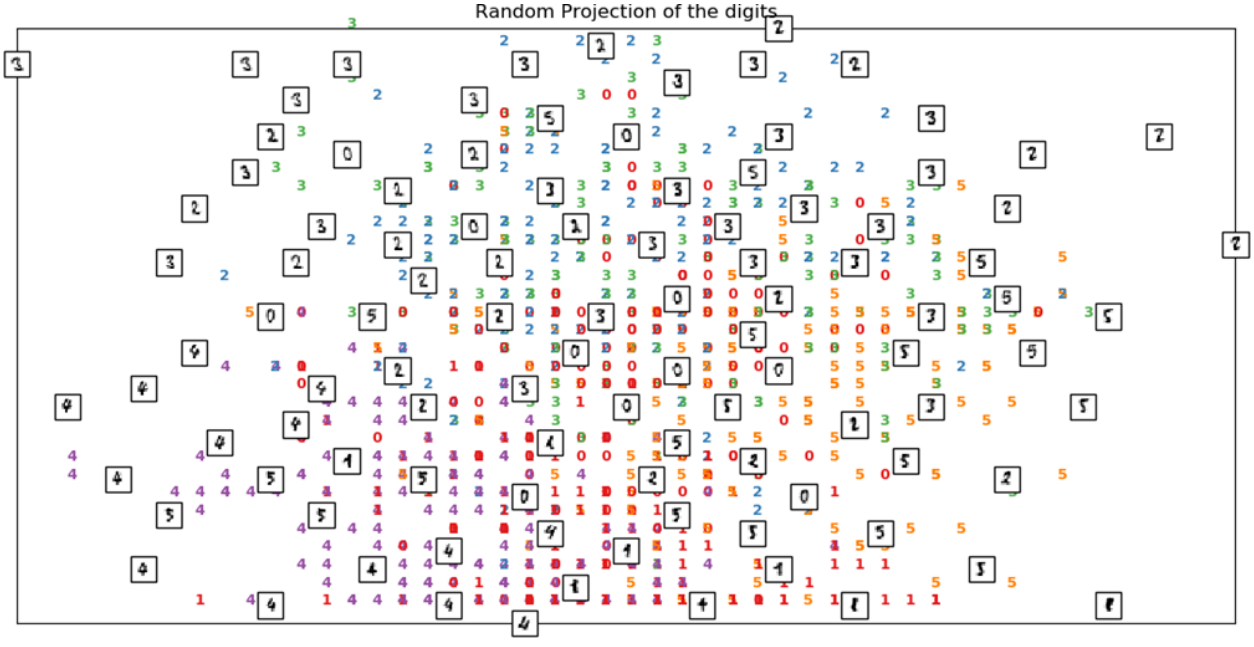
```
hasher = ensemble.RandomTreesEmbedding(n_estimators=200, random_state=0, max_depth=5)
t0 = time()
X_transformed = hasher.fit_transform(X)
pca = decomposition.TruncatedSVD(n_components=2) X_reduced =
pca.fit_transform(X_transformed)
plot_embedding(X_reduced,"Random forest embedding of the digits (time %.2fs)" %(time() -
t0)) print("Computing Spectral embedding")
embedder = manifold.SpectralEmbedding(n_components=2, random_state=0,
eigen_solver="arpack") t0 = time()
X_se = embedder.fit_transform(X)
plot_embedding(X_se,"Spectral embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing t-SNE embedding")
tsne = manifold.TSNE(n_components=2, init='pca', random_state=0) t0 = time()
X_tsne = tsne.fit_transform(X)
plot_embedding(X_tsne,"t-SNE embedding of the digits (time %.2fs)" %(time() - t0)) plt.show()
```

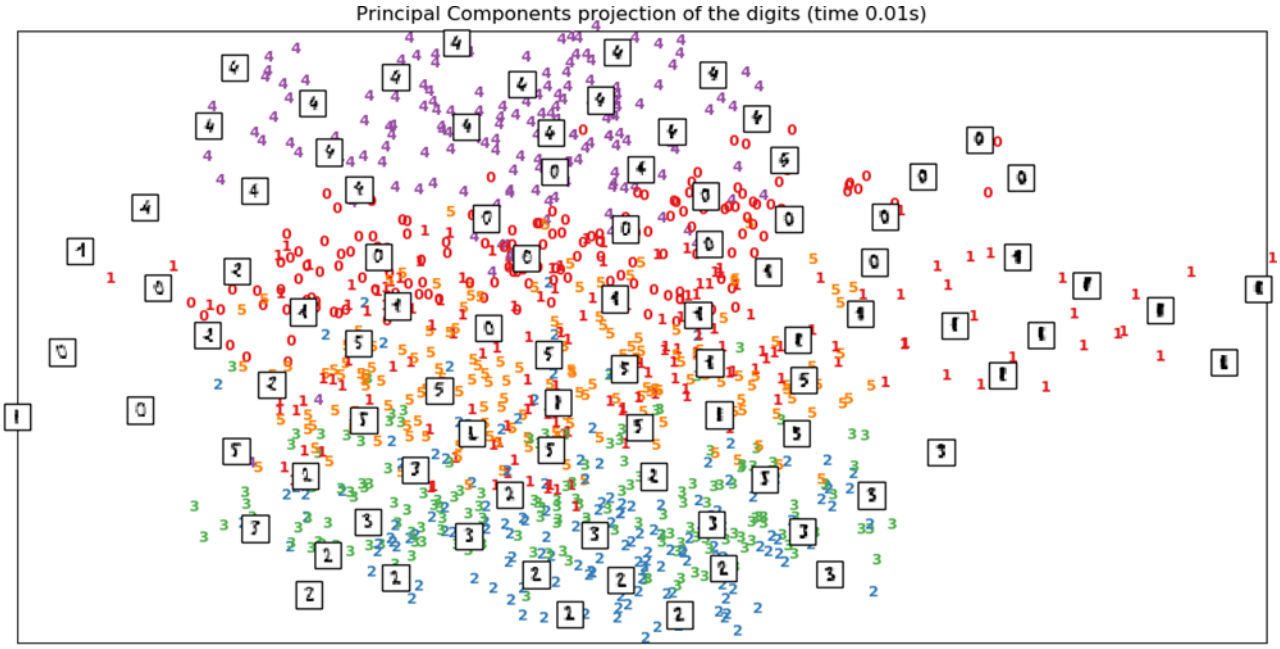
Output:

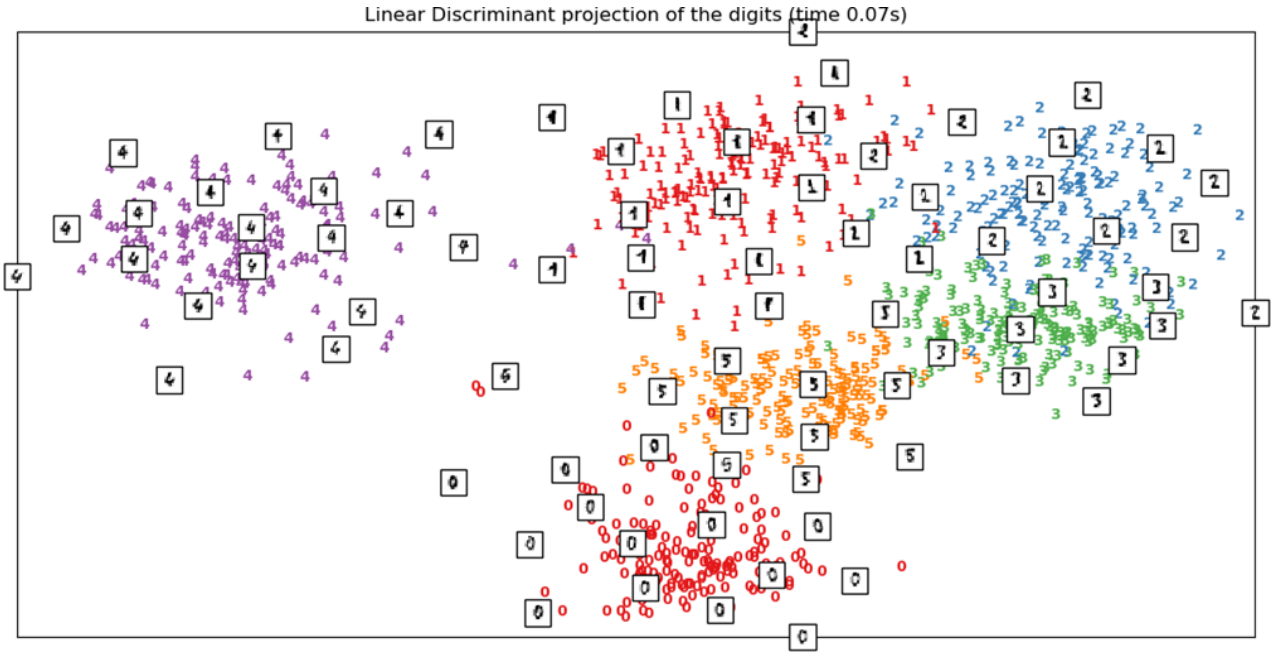


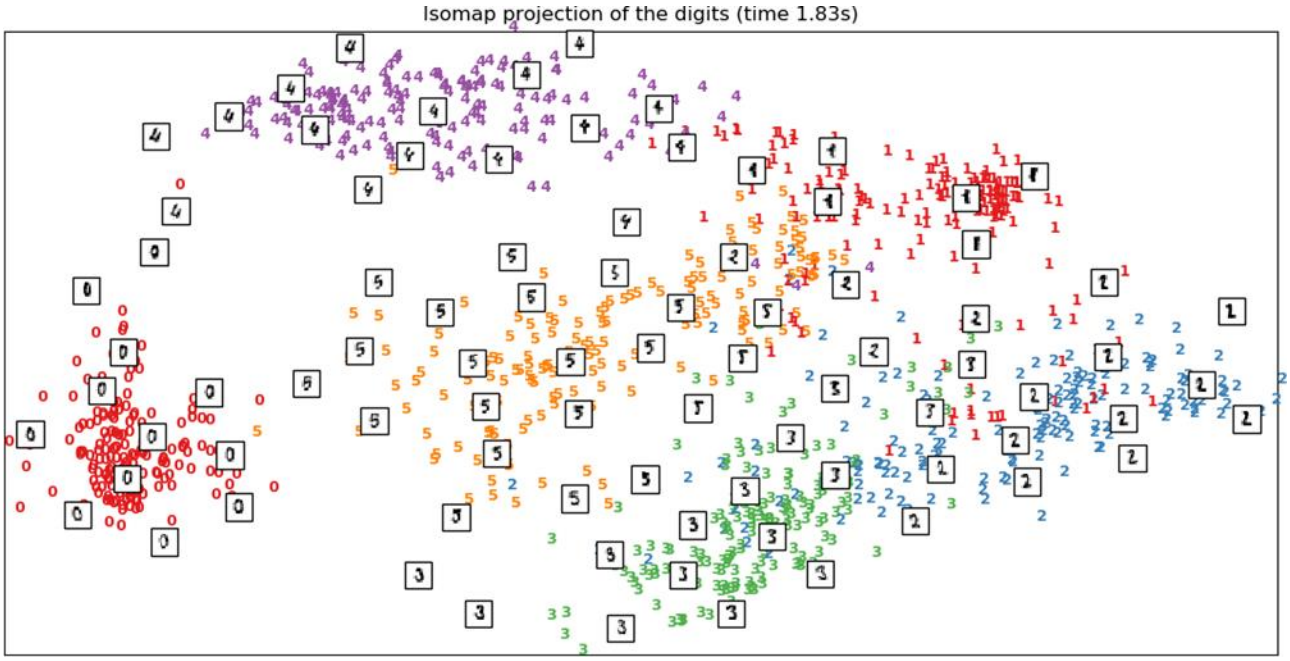
A selection from the 64-dimensional digits dataset

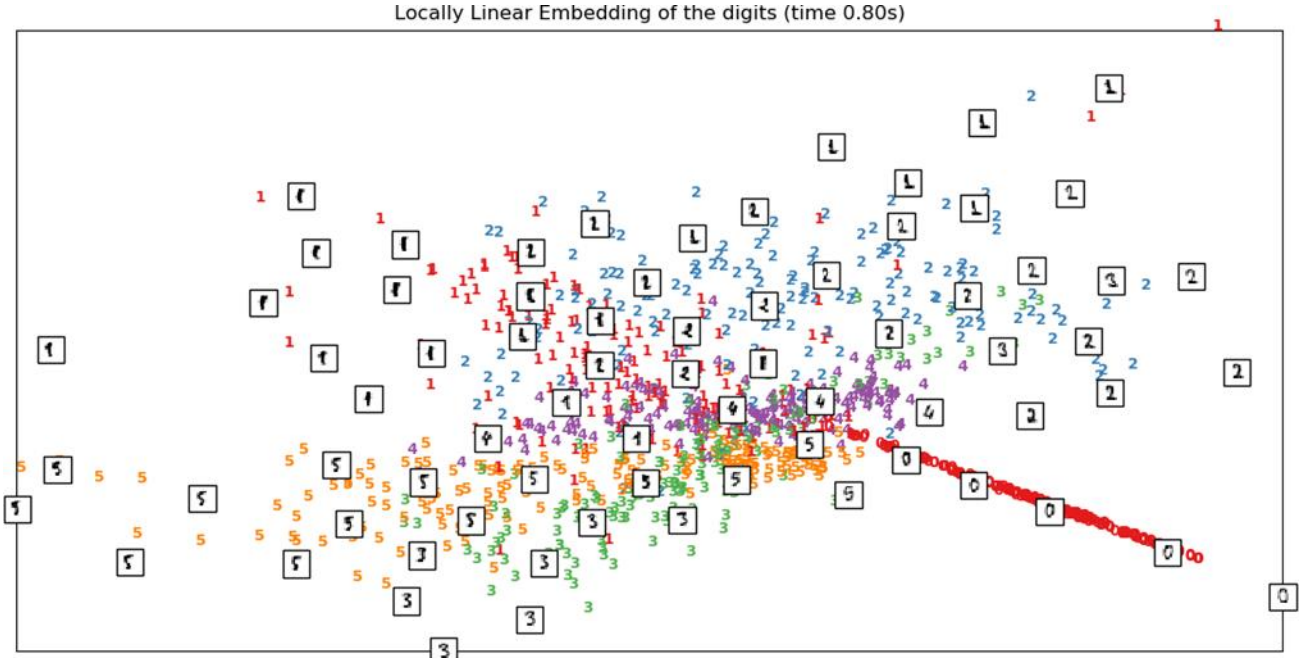
Random Projection of the digits

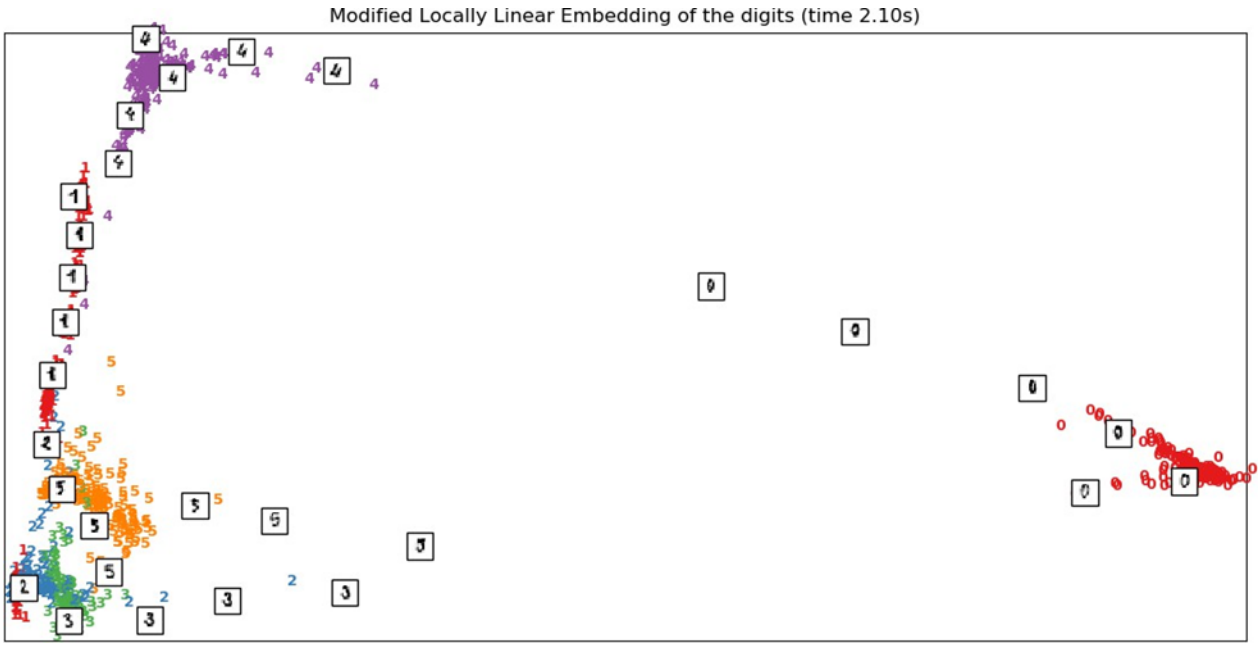


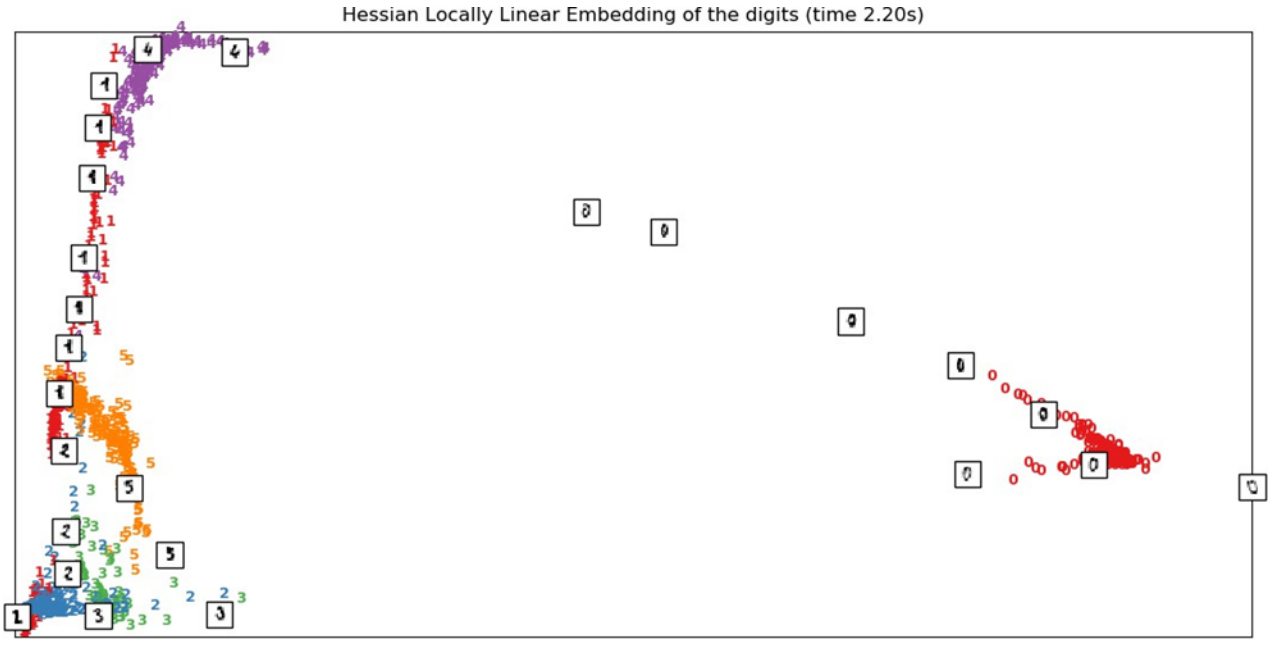

Principal Components projection of the digits (time 0.01s)

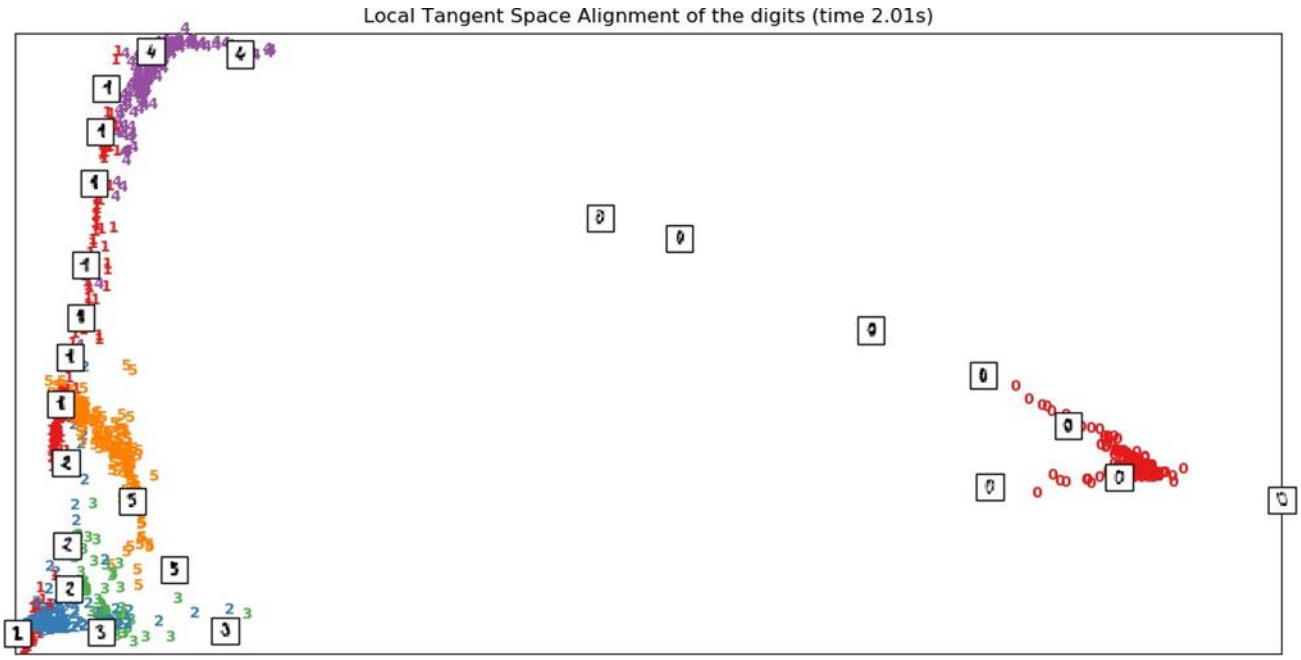

Linear Discriminant projection of the digits (time 0.07s)

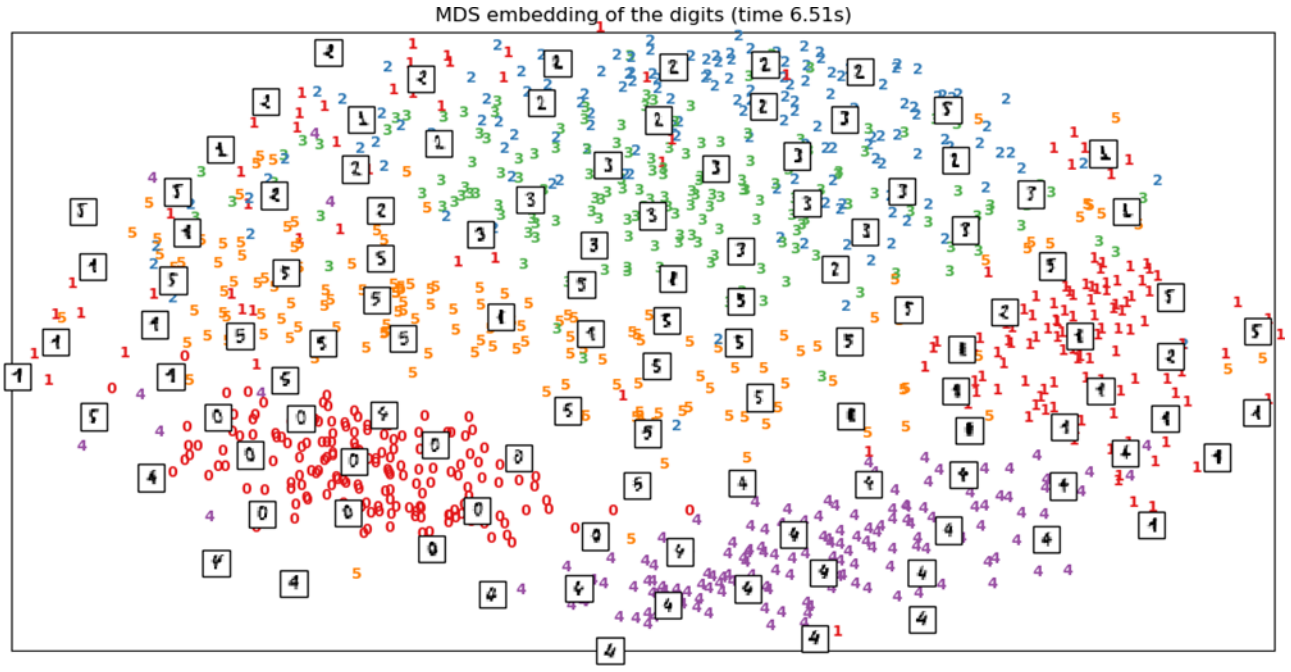

Isomap projection of the digits (time 1.83s)

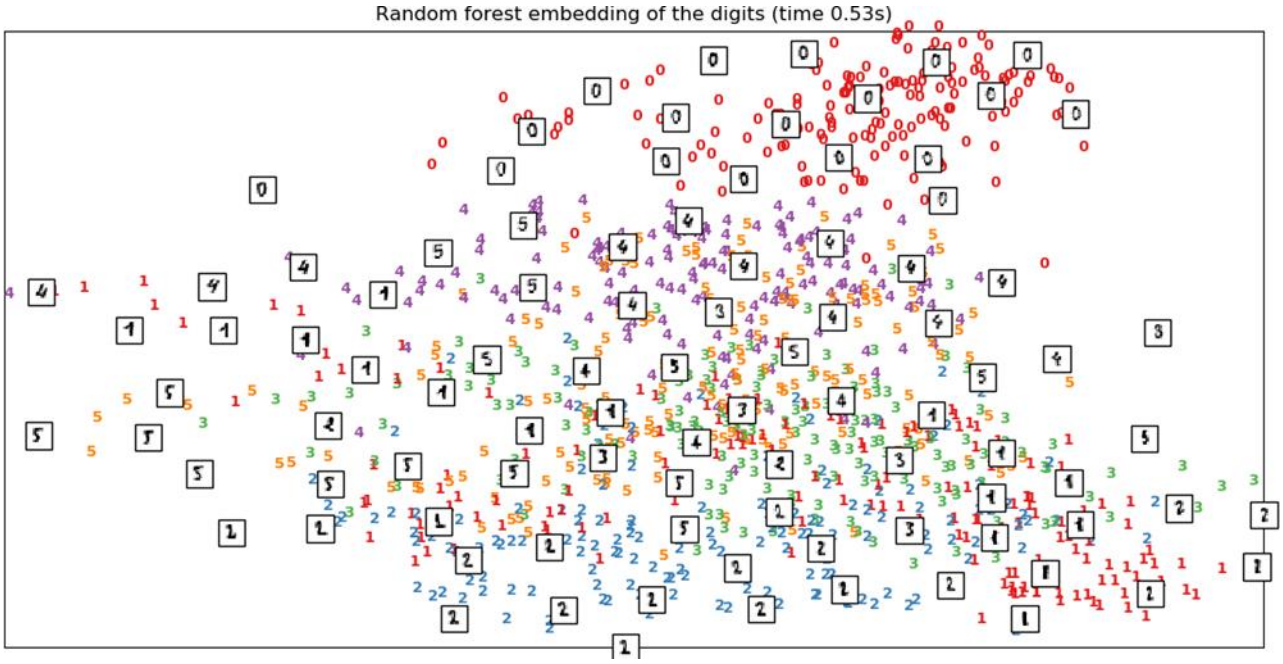

Locally Linear Embedding of the digits (time 0.80s)

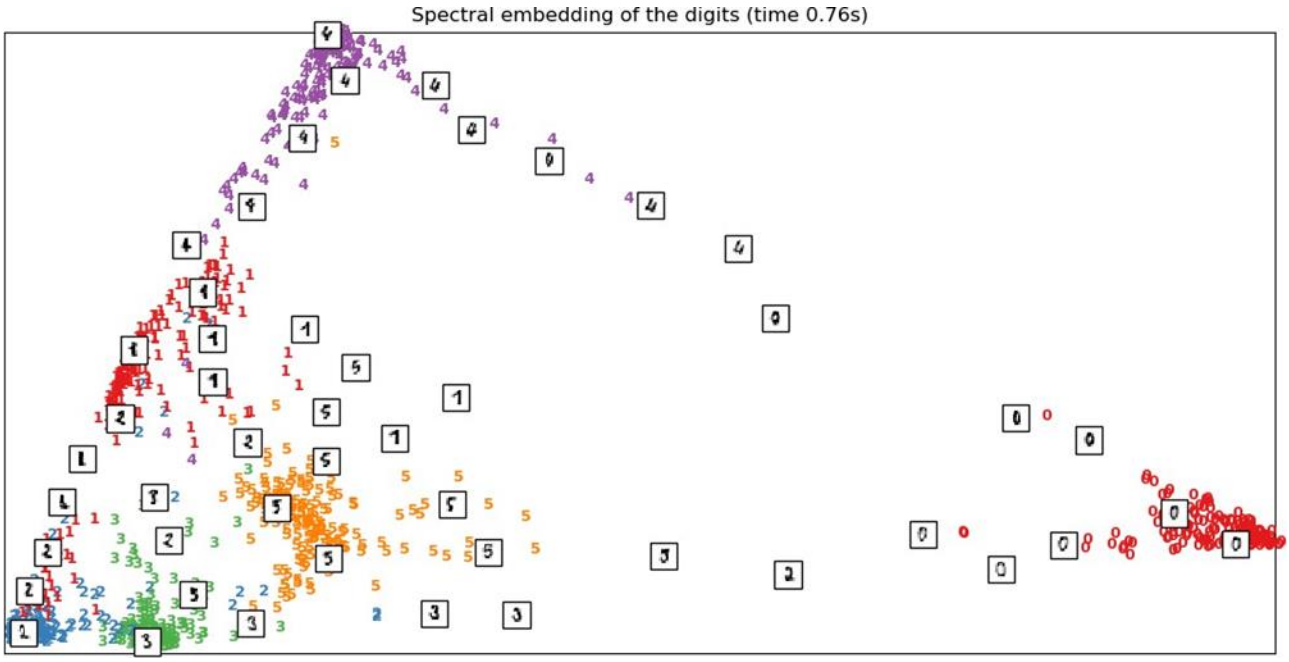

Modified Locally Linear Embedding of the digits (time 2.10s)

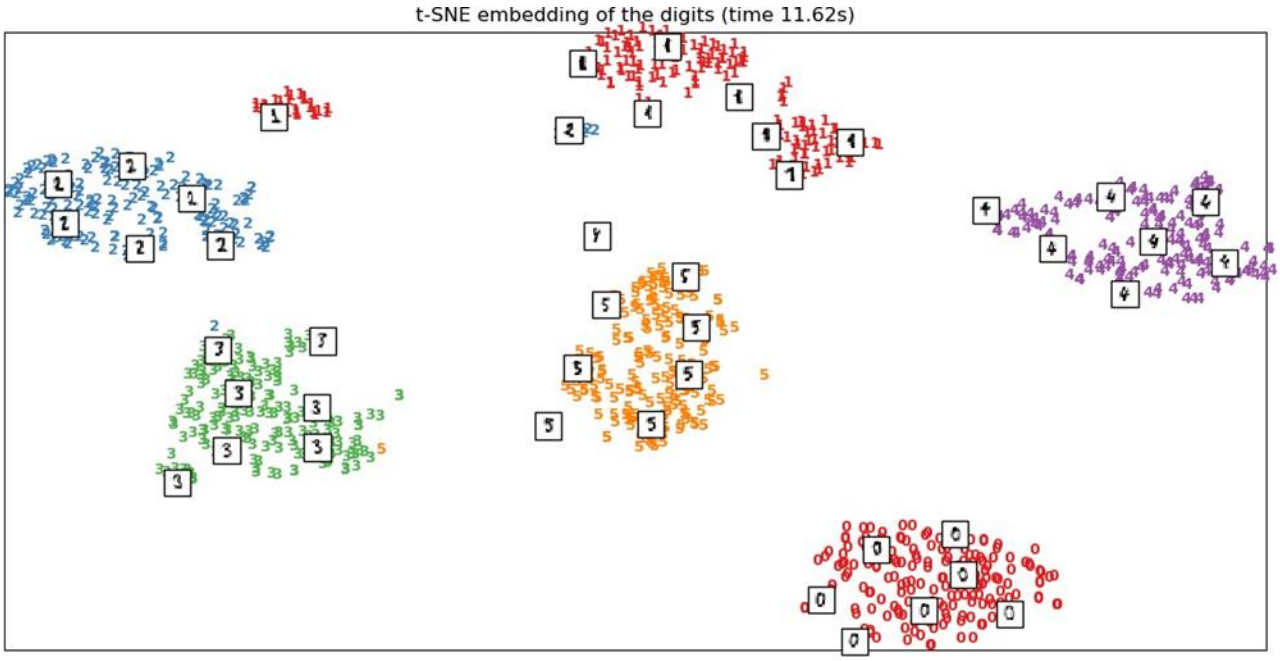

Hessian Locally Linear Embedding of the digits (time 2.20s)


Local Tangent Space Alignment of the digits (time 2.01s)


MDS embedding of the digits (time 6.51s)


Random forest embedding of the digits (time 0.53s)


Spectral embedding of the digits (time 0.76s)



You have successfully completed the container experiment. Which display format do you think is the best? The right answer is your choice, as it has to be the one that matches your own insight into the data, and there is not really a wrong answer.

Clark Ltd

Code:

```
import sys import os
import pandas as pd import sqlite3 as sq import re
from openpyxl import load_workbook 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 \
```

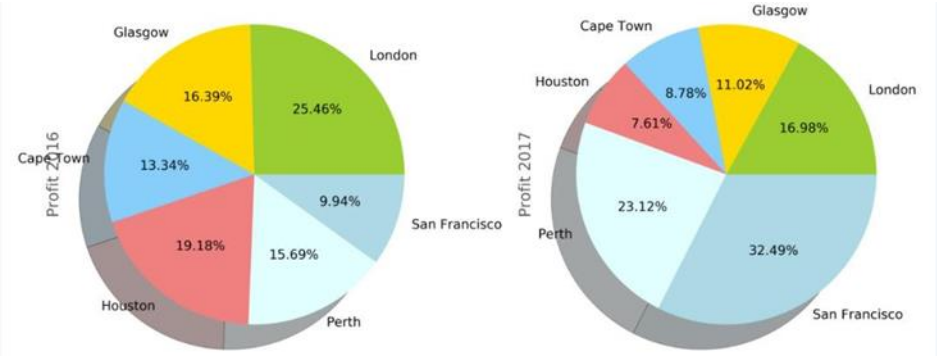
```
GROUP BY \
Year, \ Quarter, \ Country, \ Company \
HAVING \
Year=" + nYear + " \ AND \
Quarter=" + nQuarter + " \ AND \
Country='" + sCountry + "' \
AND \
Company='" + sCompany + "' \;"
sSQL=re.sub("\s\s+", " ", sSQL) sSumRaw=pd.read_sql_query(sSQL, conn) ws[sCellField] =
sSumRaw["nSumTotal"][0] * nSumSign print('Set cell',sCellField,' to ', sSumField,'Total')
wb.save(sFileName)
```

Output:

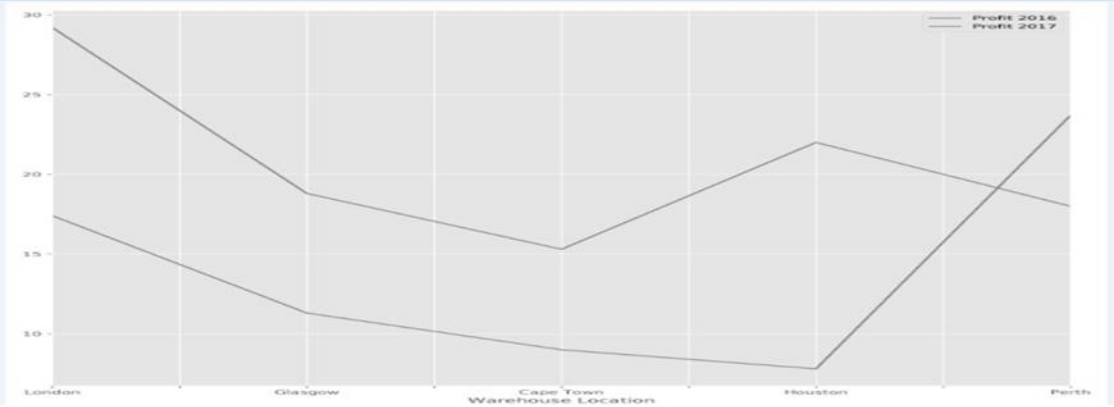
You now have all the reports you need.
Check the Following files for generated reports in C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/

1. Report-Balance-Sheet-2000Q1-Clark-Afghanistan.xlsx
2. Report-Balance-Sheet-2000Q1-Hillman-Afghanistan.xlsx
3. Report-Balance-Sheet-2000Q1-Krennwallner-Afghanistan.xlsx
4. Report-Balance-Sheet-2000Q1-Vermeulen-Afghanistan.xlsx
5. Report-Balance-Sheet-2000Q1-Clark-AlandIslands.xlsx

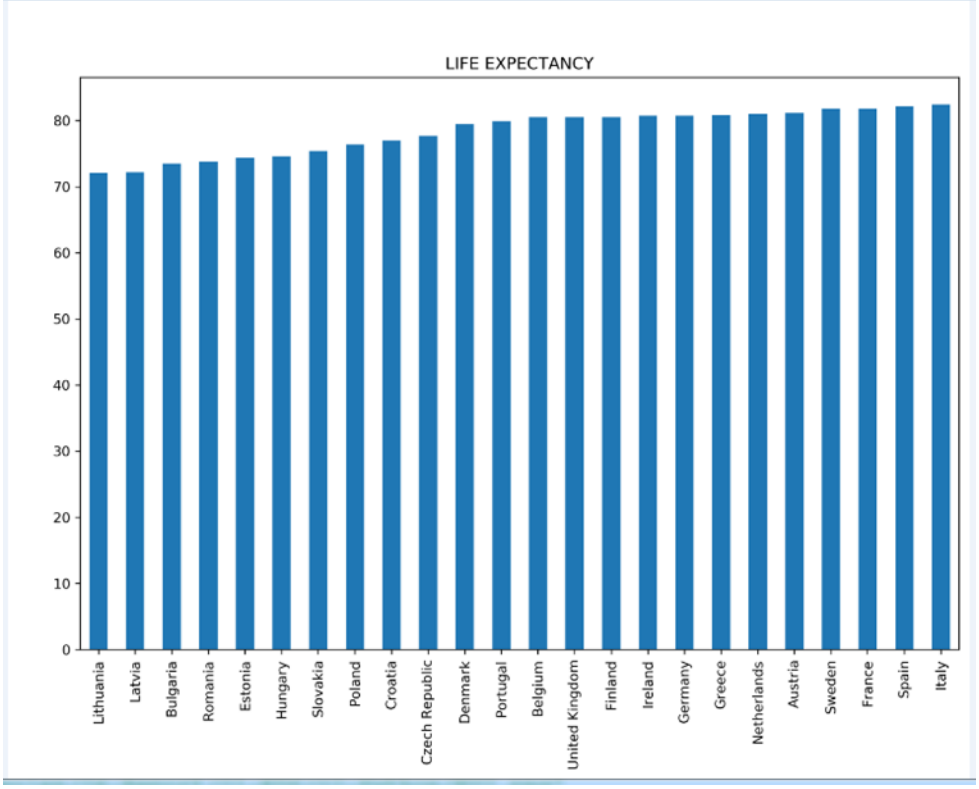
Pie Graph Double Pie
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py



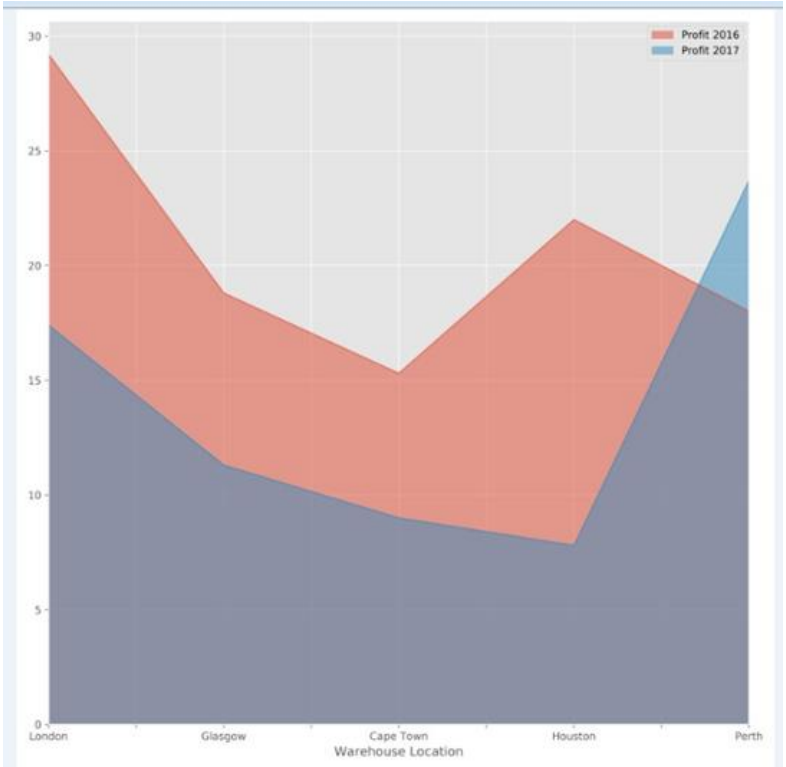
Line Graph
C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py



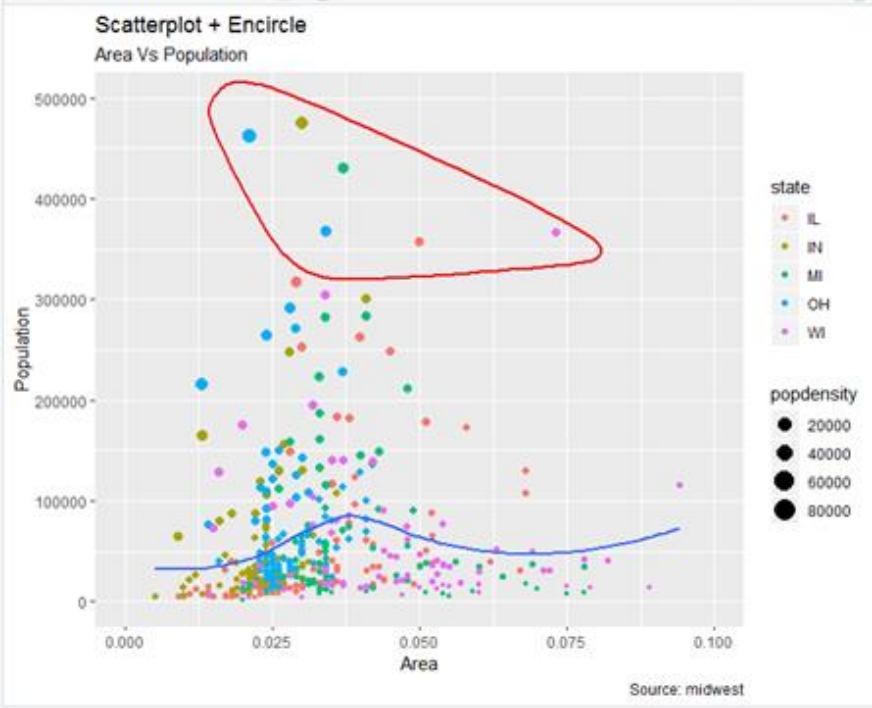
Bar Graph / Horizontal Bar Graph
C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py



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

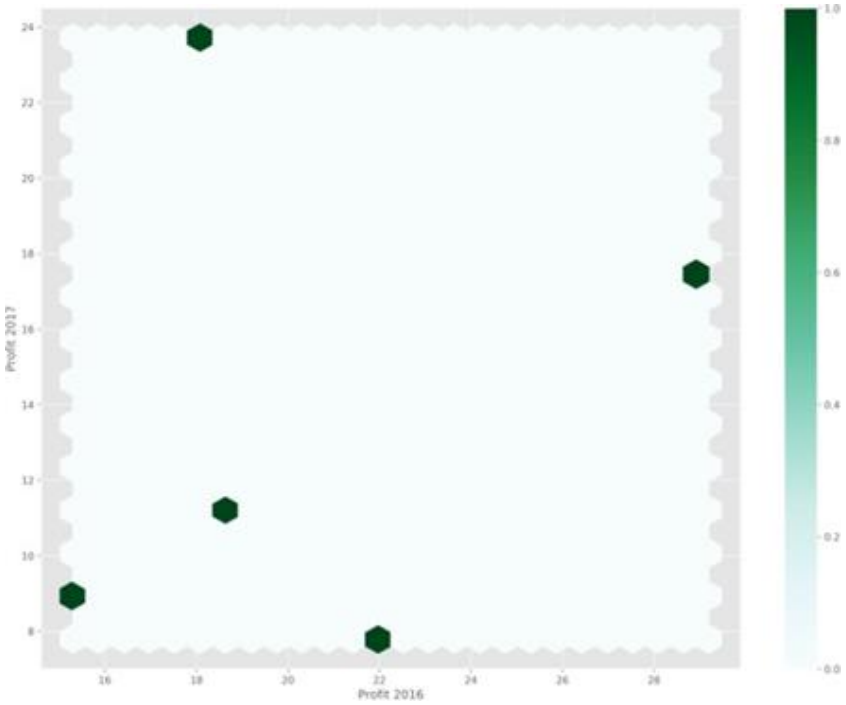


Scatter Graph : VKHCG/03-Hillman/06-Report/Report-Scatterplot-With-Encircling.r



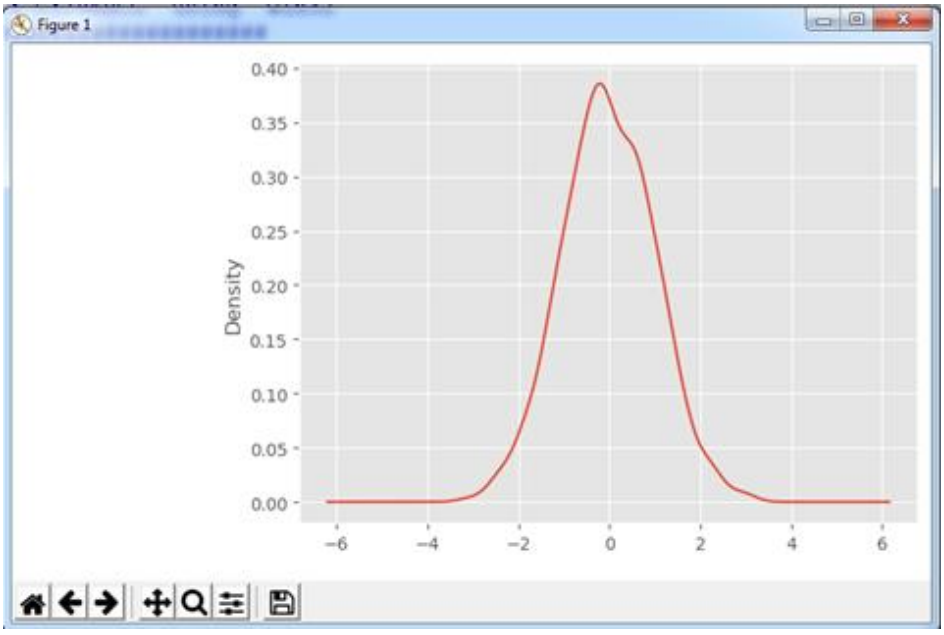
Hexbin:

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



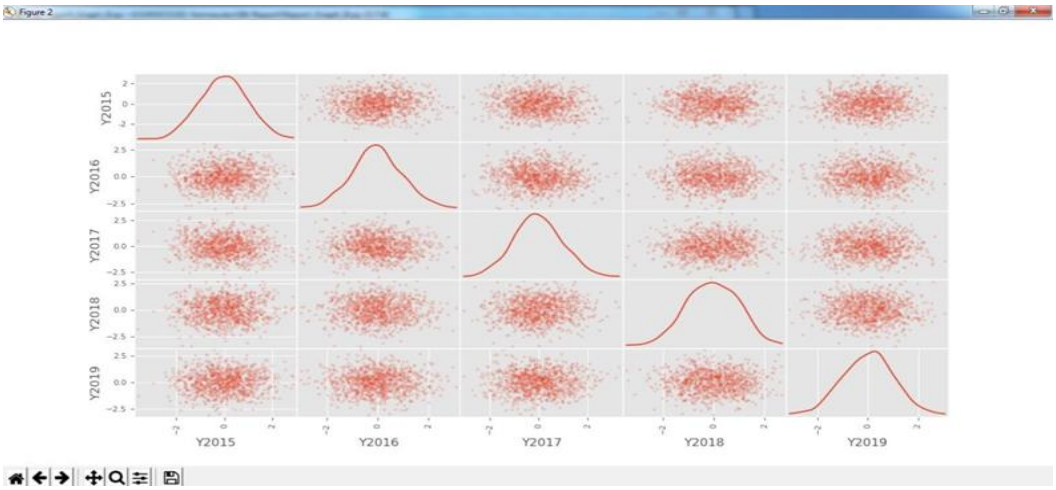
Kernel Density Estimation (KDE) Graph

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



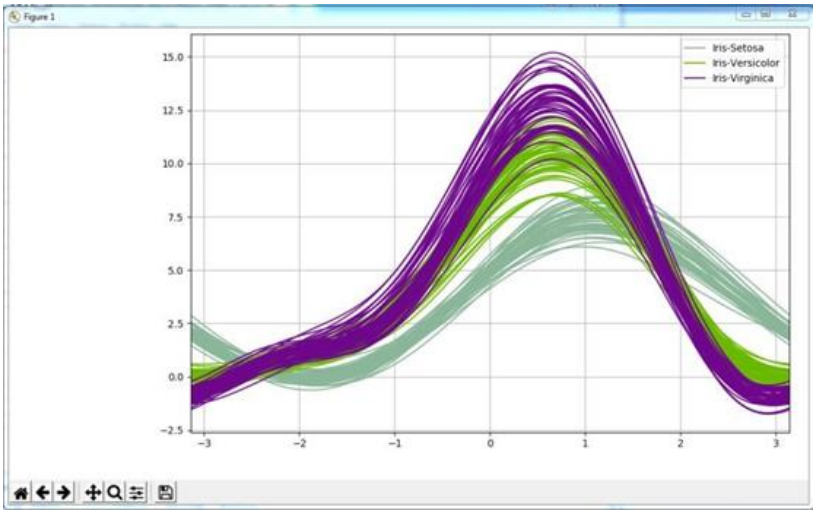
Scatter Matrix Graph

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



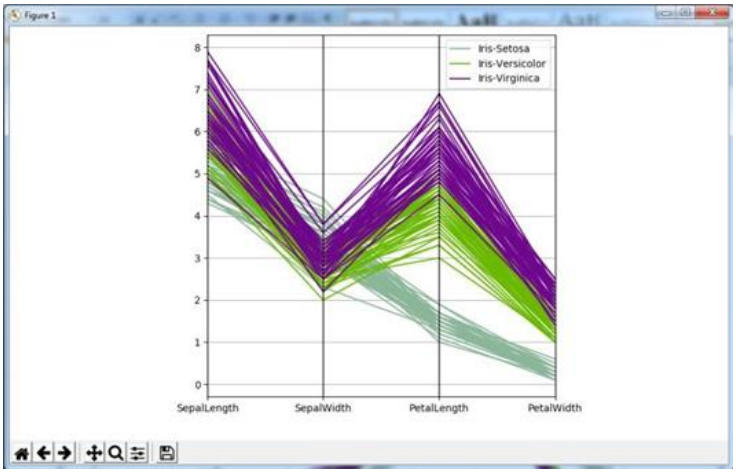
Andrews' Curves

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



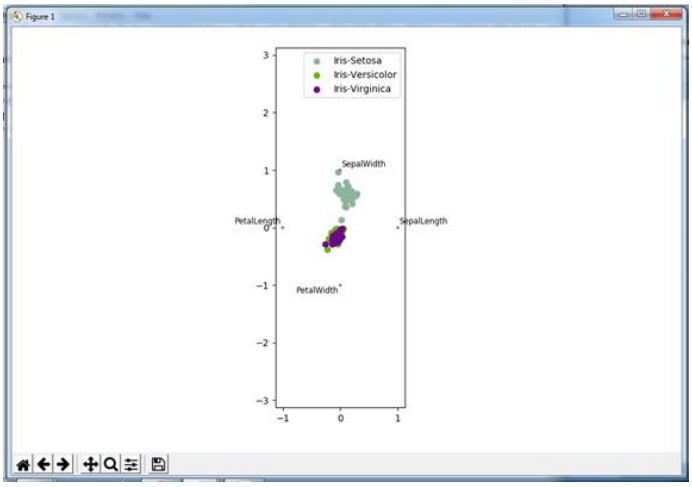
Parallel Coordinates

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



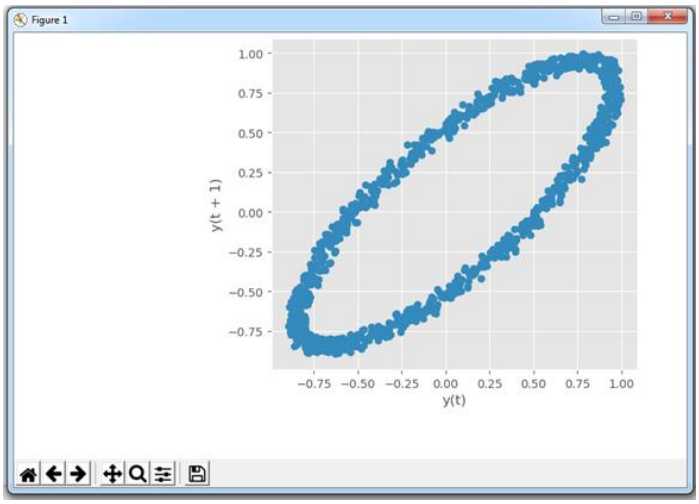
RADVIZ Method

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



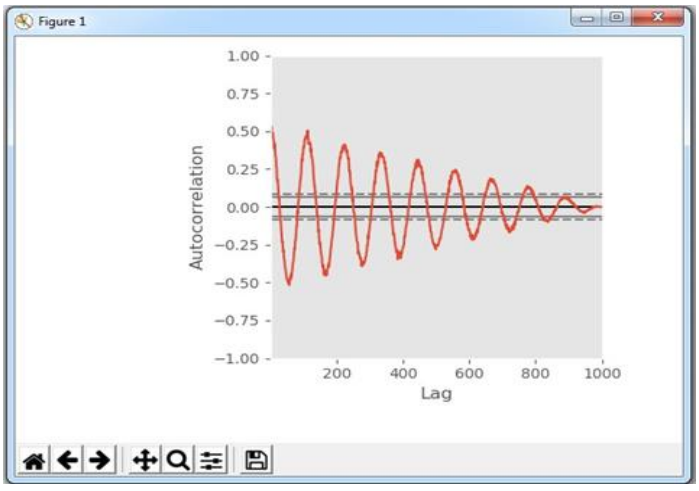
Lag Plot

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



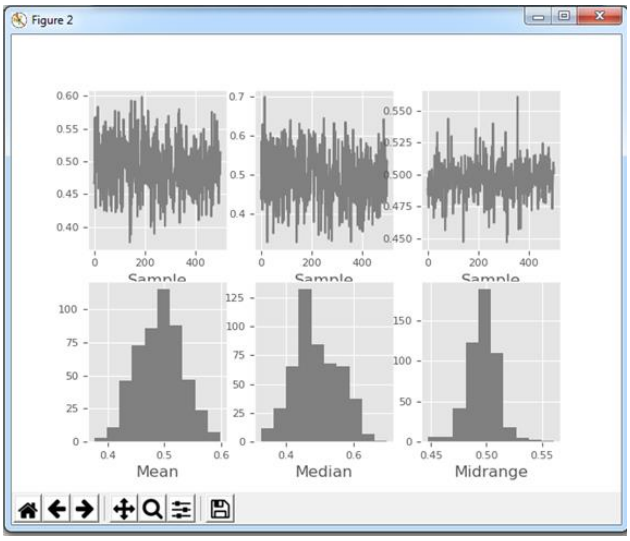
Autocorrelation Plot

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



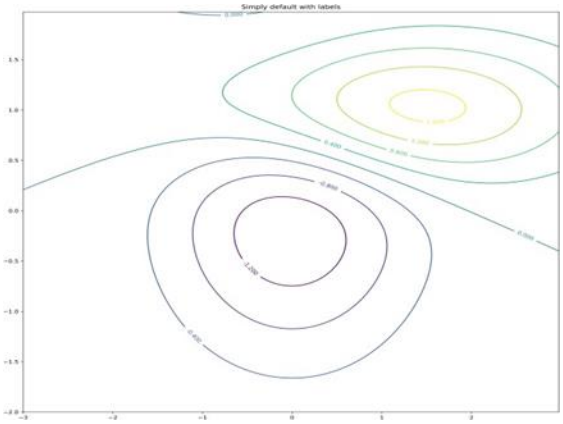
Bootstrap Plot

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



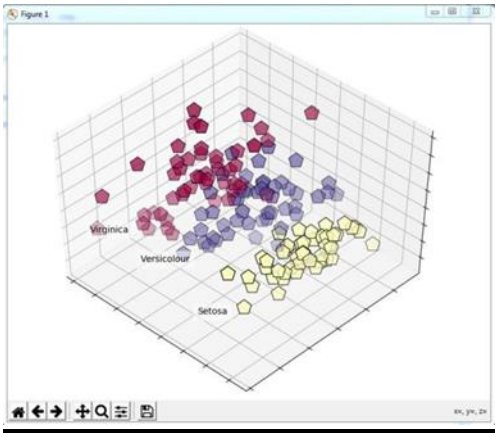
Contour Graphs

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



3D Graphs

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

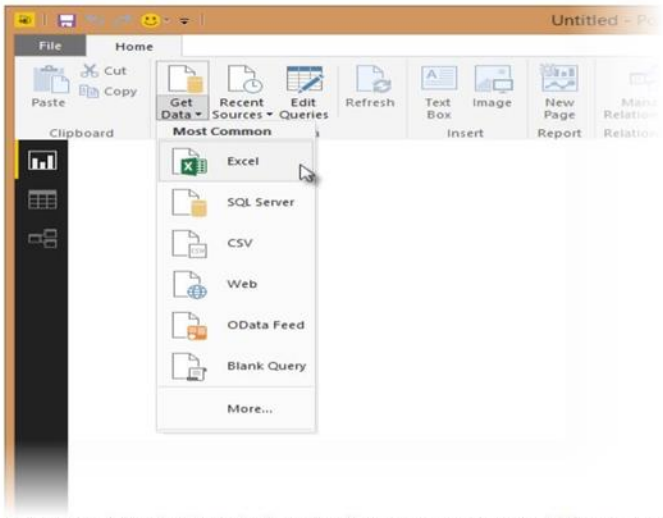


Practical No 9: Data Visualization with Power BI

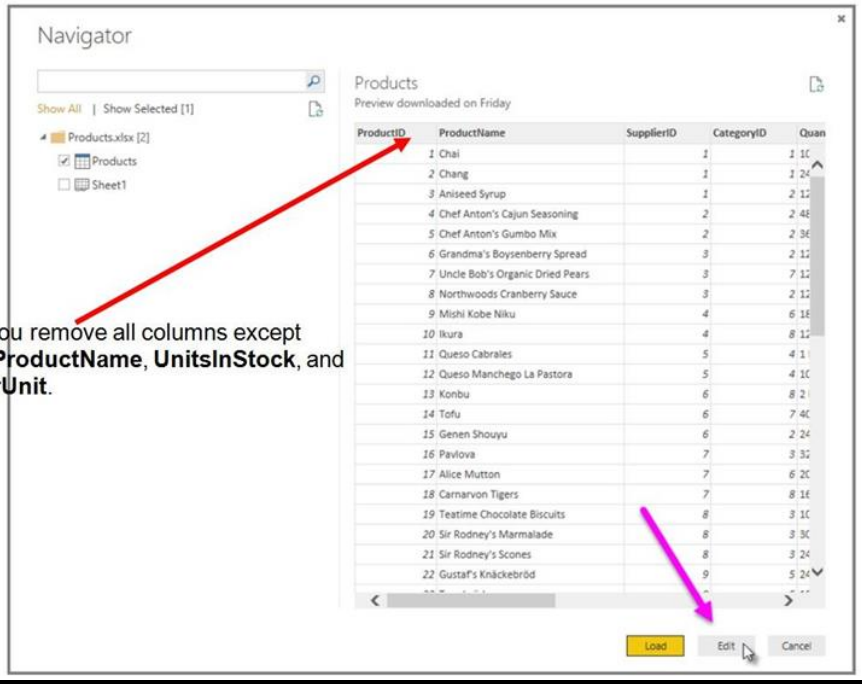
Case Study : Sales Data

Step 1: Connect to an Excel workbook

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



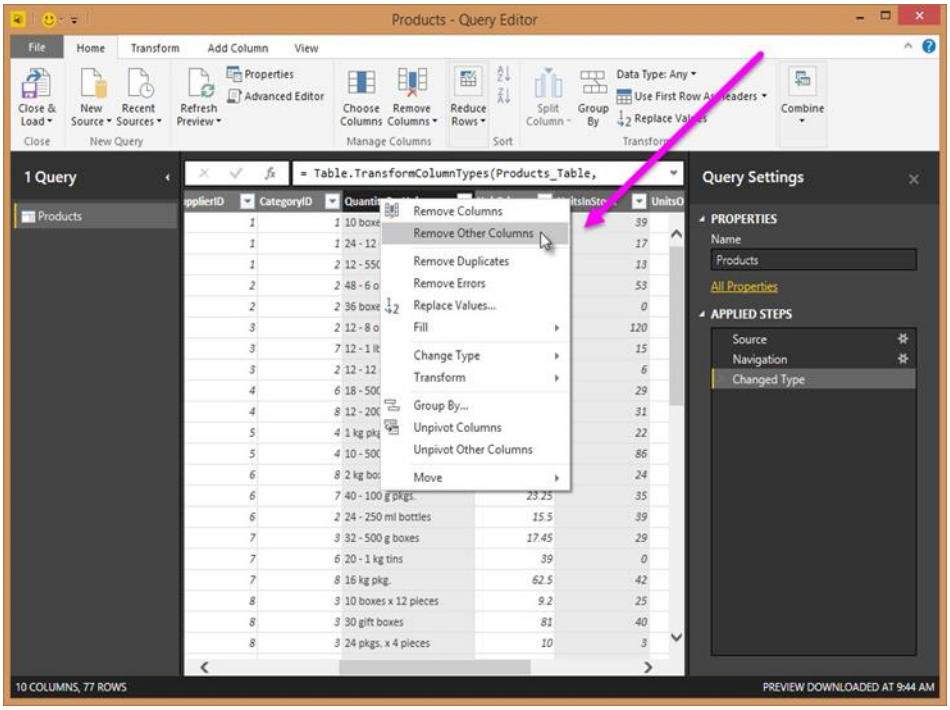
3. If you select the Get Data button directly, you can also select **File > Excel** and select **Connect**.
4. In the **Open File** dialog box, select the **Products.xlsx** file.

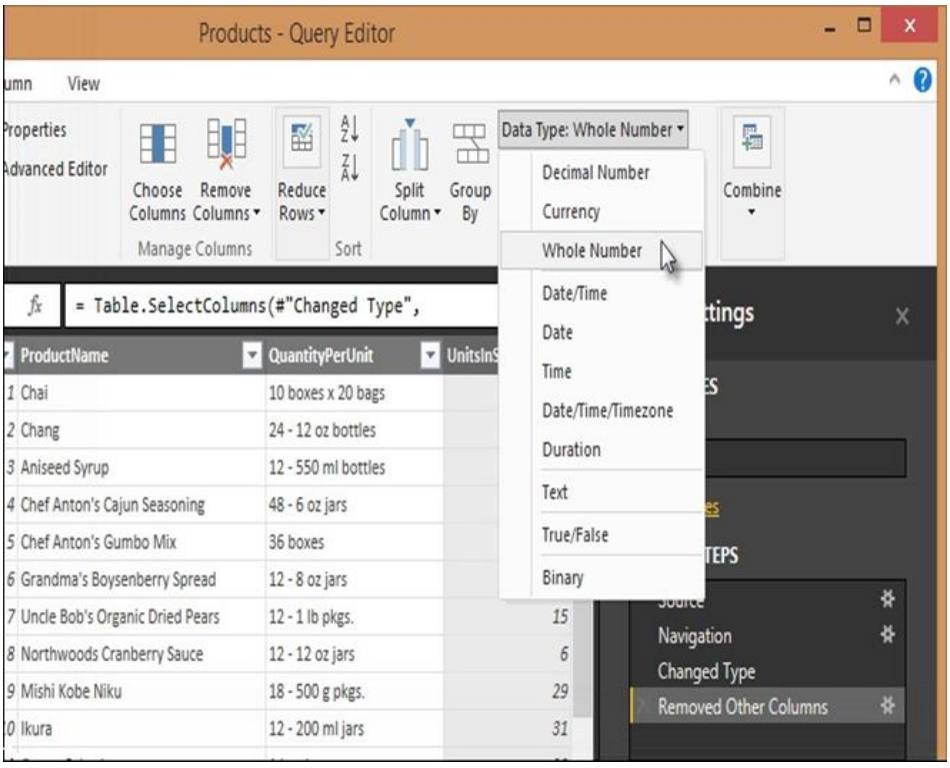


In this step you remove all columns except **ProductID**, **ProductName**, **UnitsInStock**, and **QuantityPerUnit**.

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

1. Query Editor, select the ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns (use Ctrl+Click to select more than one column, or Shift+Click to select columns that are beside each other)
2. Select Remove Columns□ Remove Other Columns from the ribbon, or right-click on a column header and click Remove Other Columns.





3. Change the data type of the UnitsInStock column

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

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

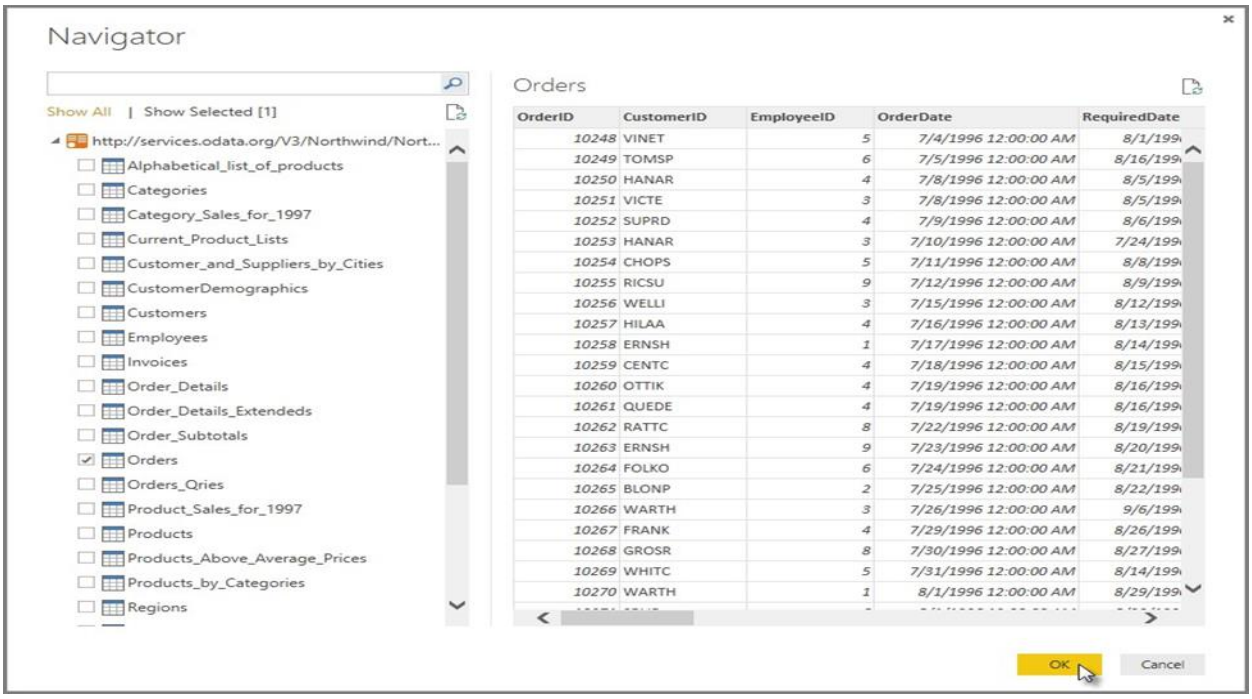
Task 2: Import order data from an OData feed

You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below:

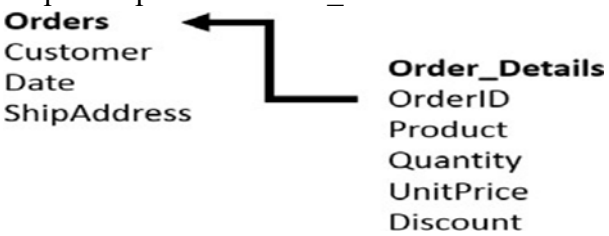
http://services.odata.org/V3/Northwind/Northwind.svc/

Step 1: Connect to an OData feed

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



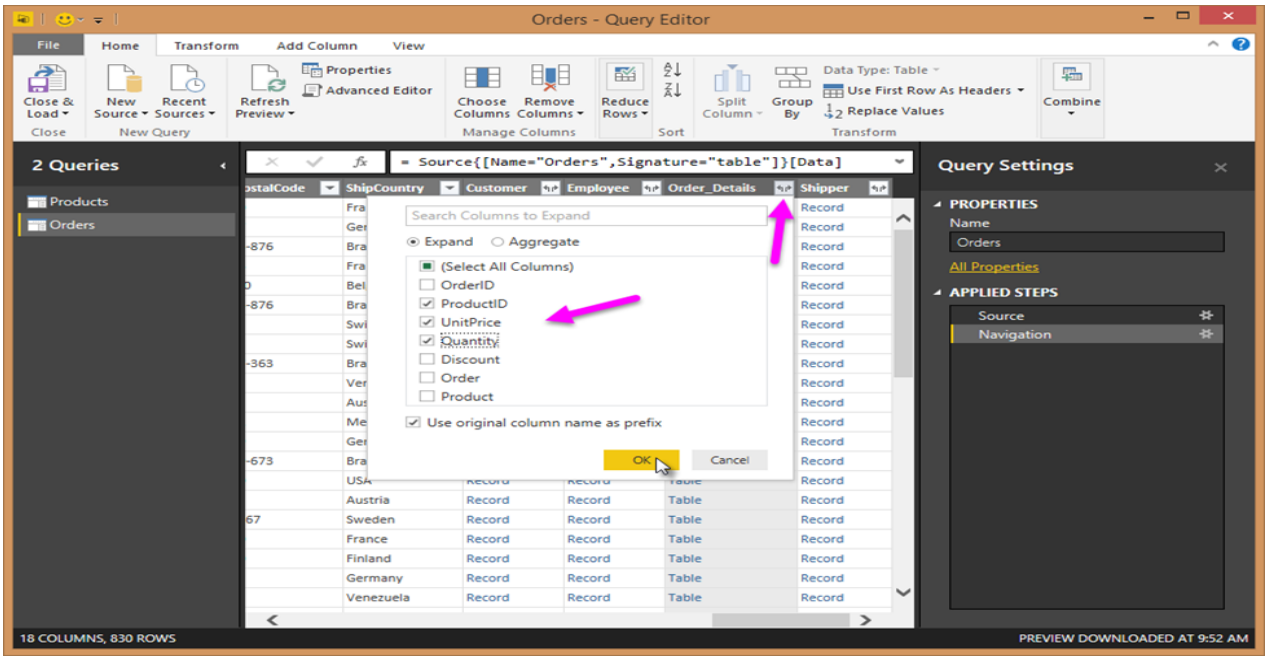
Step 2: Expand the Order_Details table



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

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

click OK.



Step 3: Remove other columns to only display columns of interest
In this step you remove all columns except OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and Order_Details.Quantity columns. In the previous task, you used Remove Other Columns. For this task, you remove selected columns.
In the Query View, select all columns by completing

a) Click the first column (OrderID).
b) Shift+Click the last column (Shipper).
c) Now that all columns are selected, use Ctrl+Click to unselect the following columns: OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and Order_Details.Quantity.

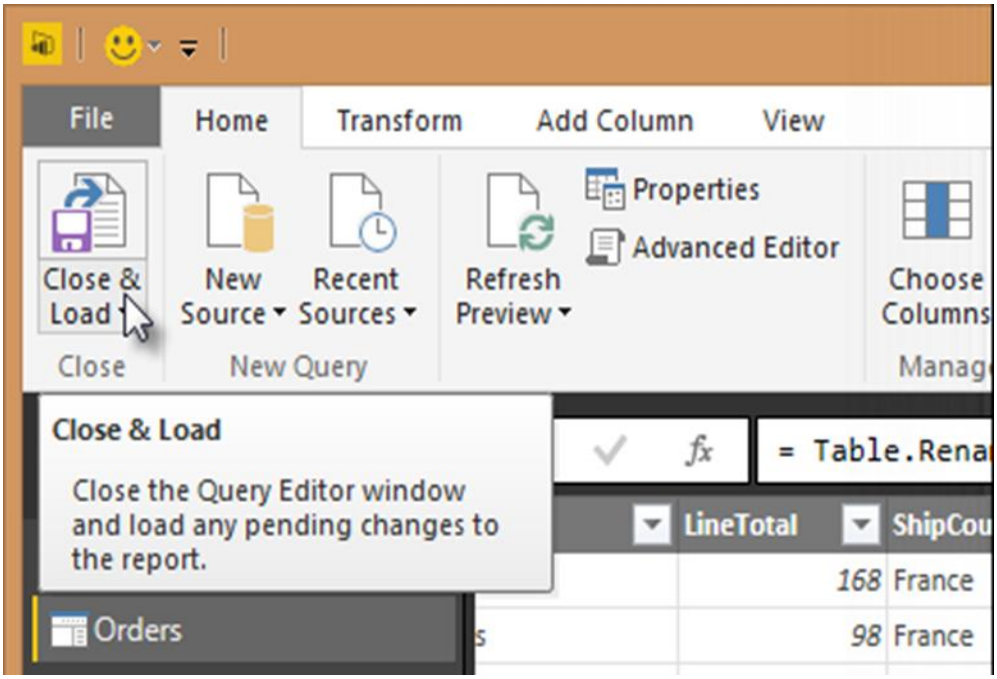
Now that only the columns we want to remove are selected, right-click on any selected column header and click Remove Columns.

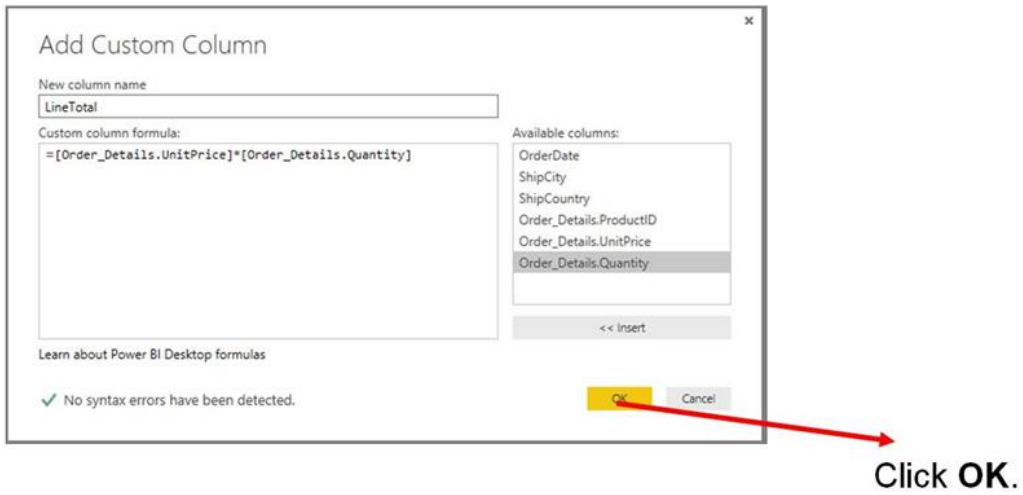
Step 4: Calculate the line total for each Order_Details row

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

Calculate the line total for each Order_Details row:

1. In the Add Column ribbon tab, click Add Custom Column.
2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order_Details.UnitPrice] * [Order_Details.Quantity].
3. In the New column name textbox, enter LineTotal.

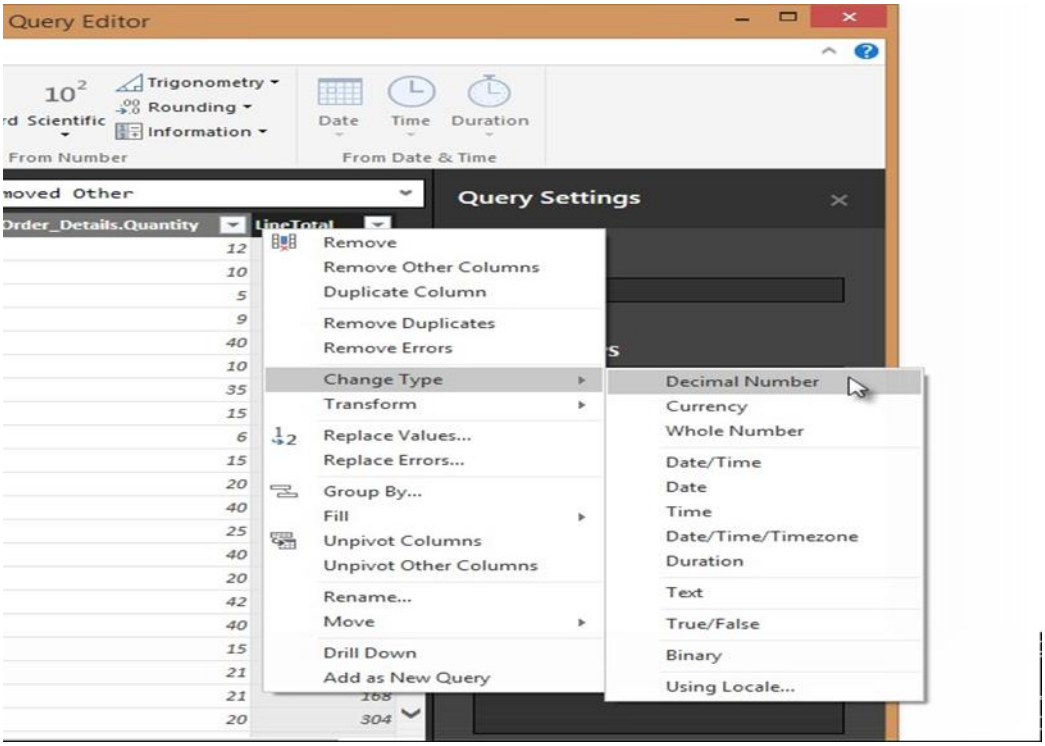




Step 5: Set the datatype of the LineTotal field

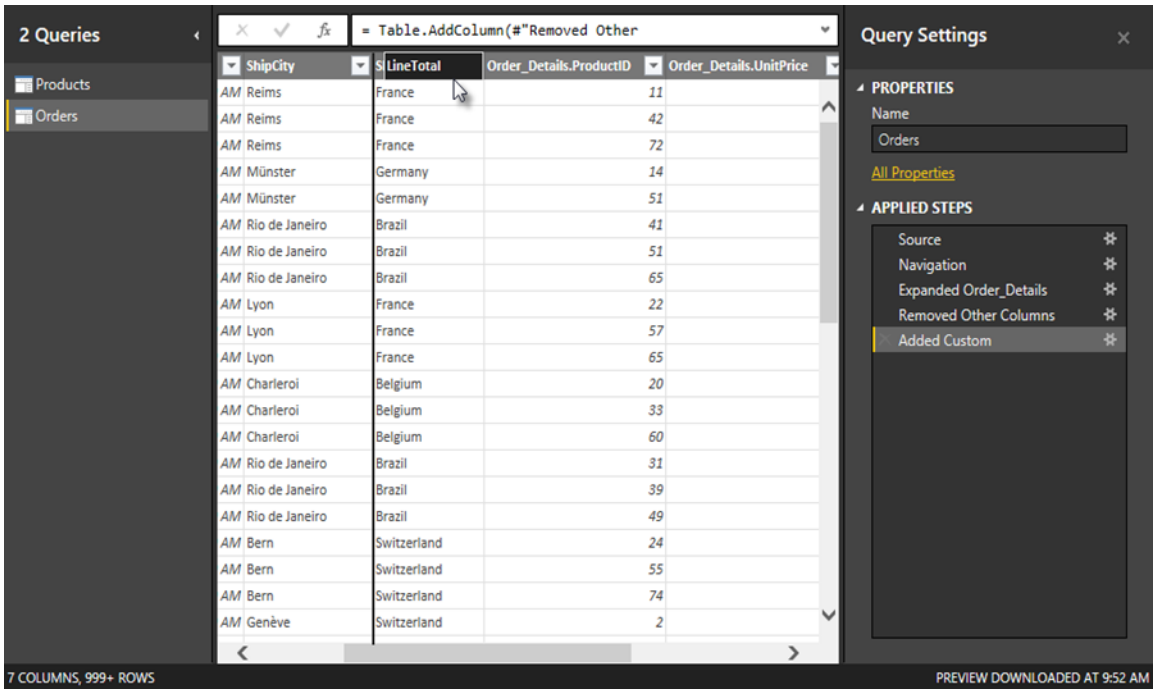
Right click the **LineTotal** column.

2. Select Change Type and choose Decimal Number.

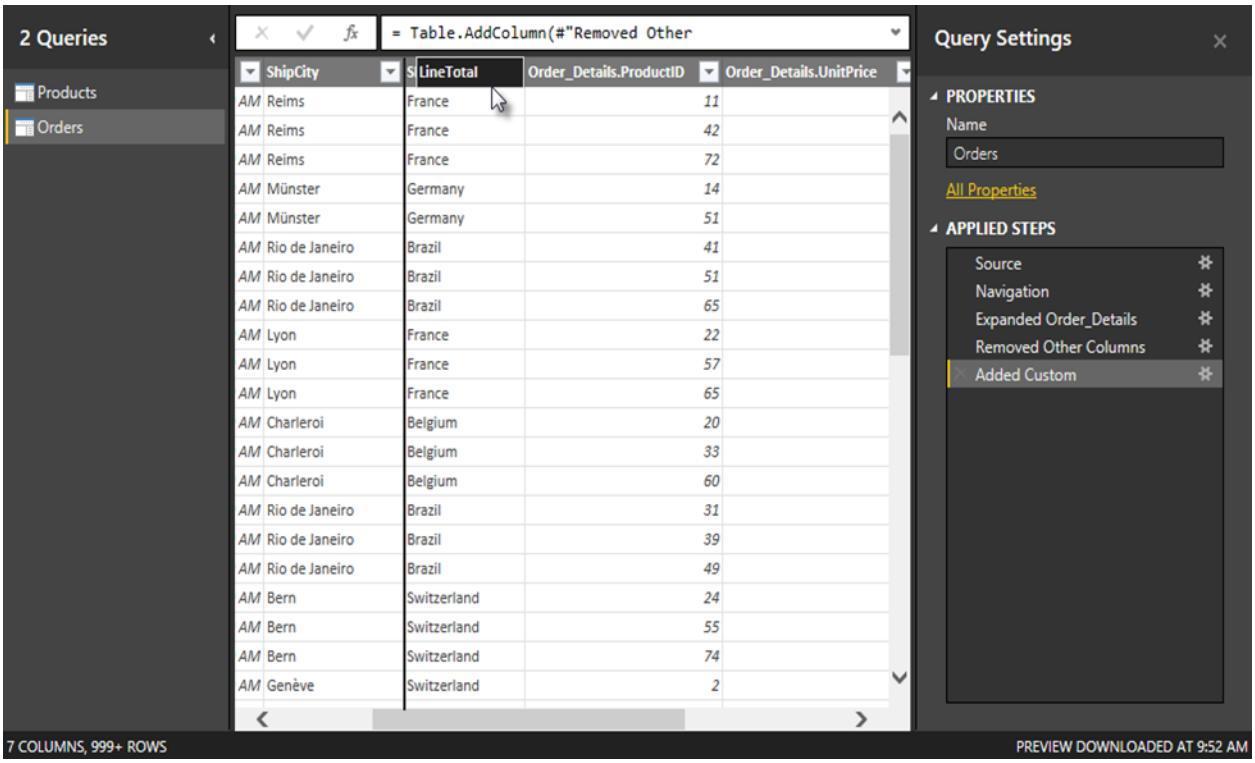


Step 6: Rename and reorder columns in the query

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



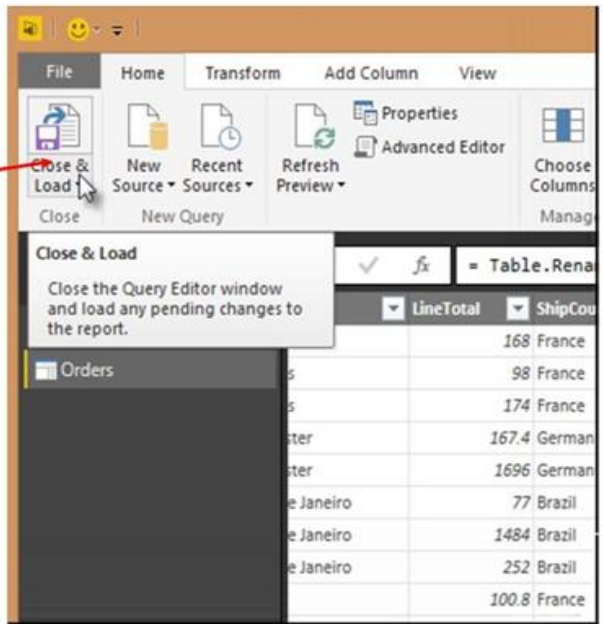
2. Remove the Order_Details. prefix from the Order_Details.ProductID, Order_Details.UnitPrice and Order_Details.Quantity columns, by double-clicking on each column header, and then deleting that text from the column name.



Task 3: Combine the Products and Total Sales queries

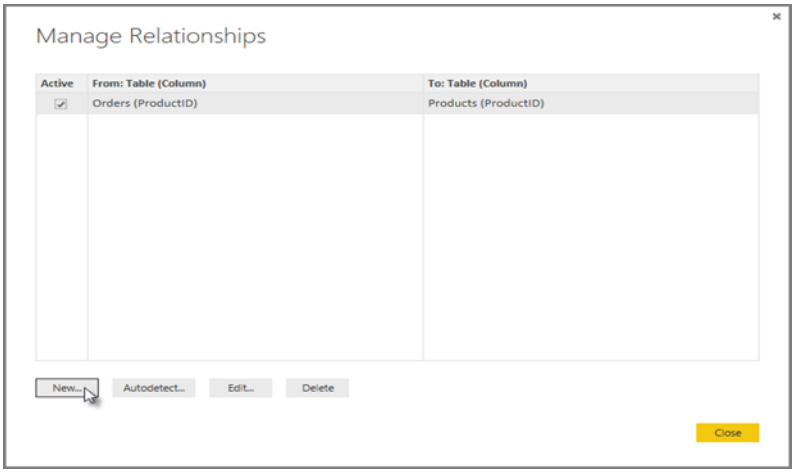
Step 1: Confirm the relationship between Products and Total Sales

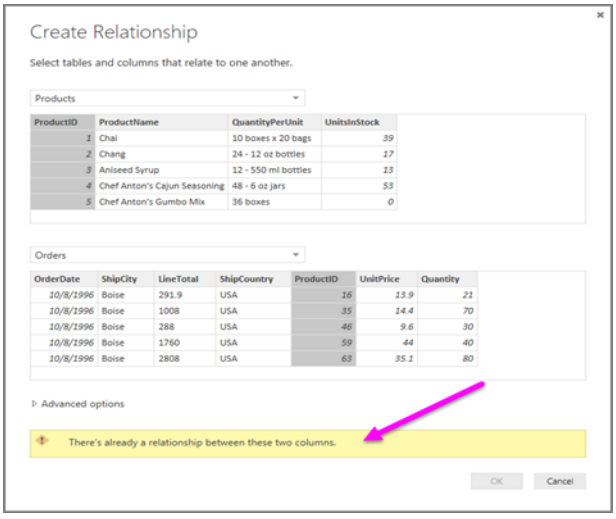
1. First, we need to load the model that we created in Query Editor into Power BI Desktop. From the **Home** ribbon of Query Editor, select **Close & Load**.



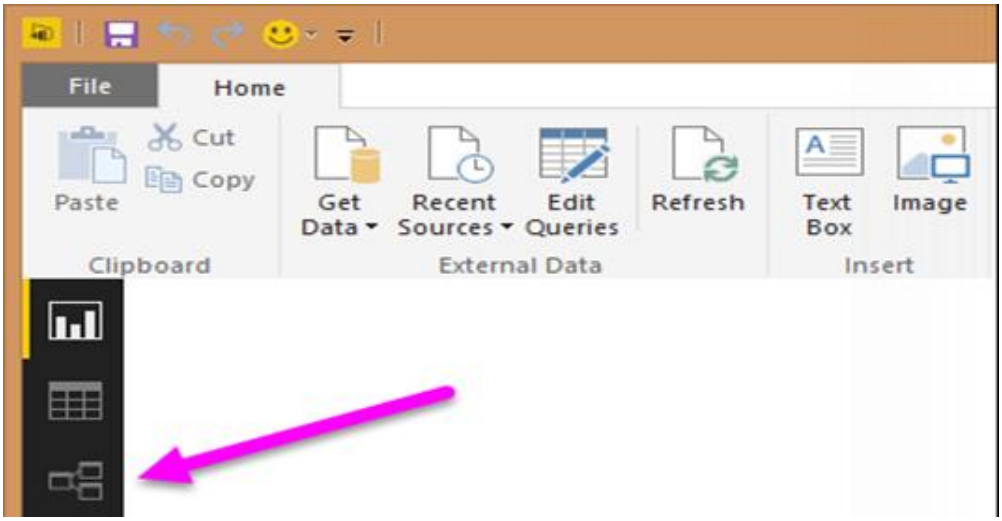
Power BI Desktop loads the data from the two queries

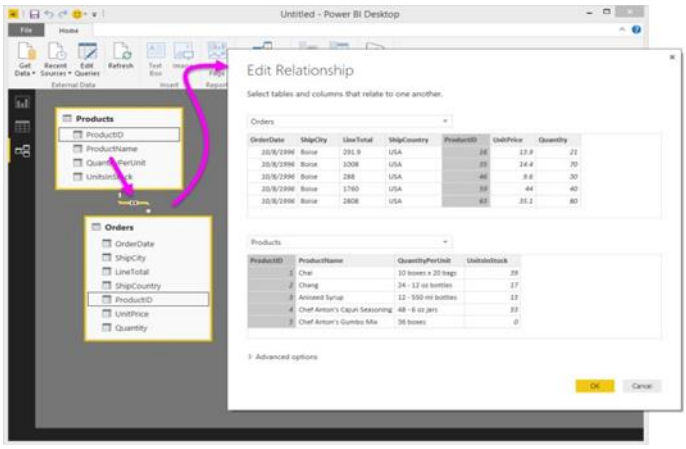
1. Once the data is loaded, select the Manage Relationships button Home ribbon
2. Select the New… button
3. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.





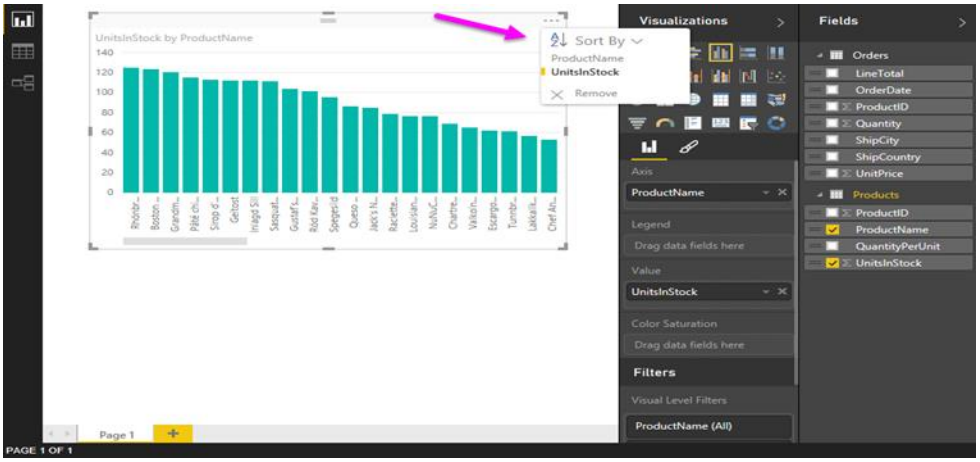
4. Select Cancel, and then select Relationship view in Power BI Desktop.

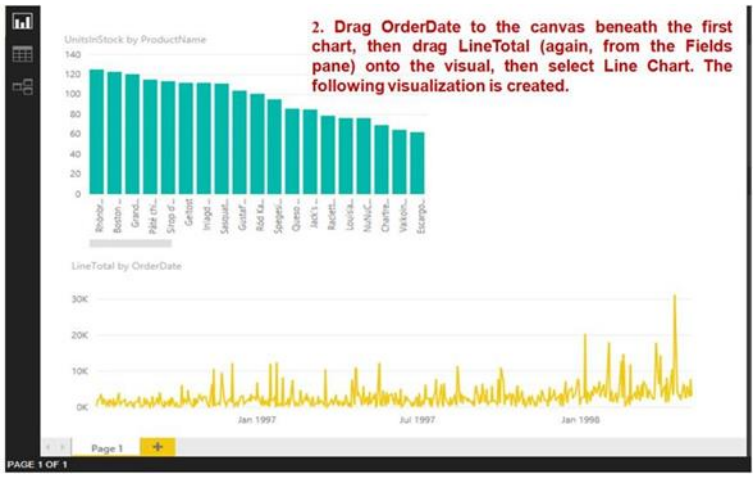




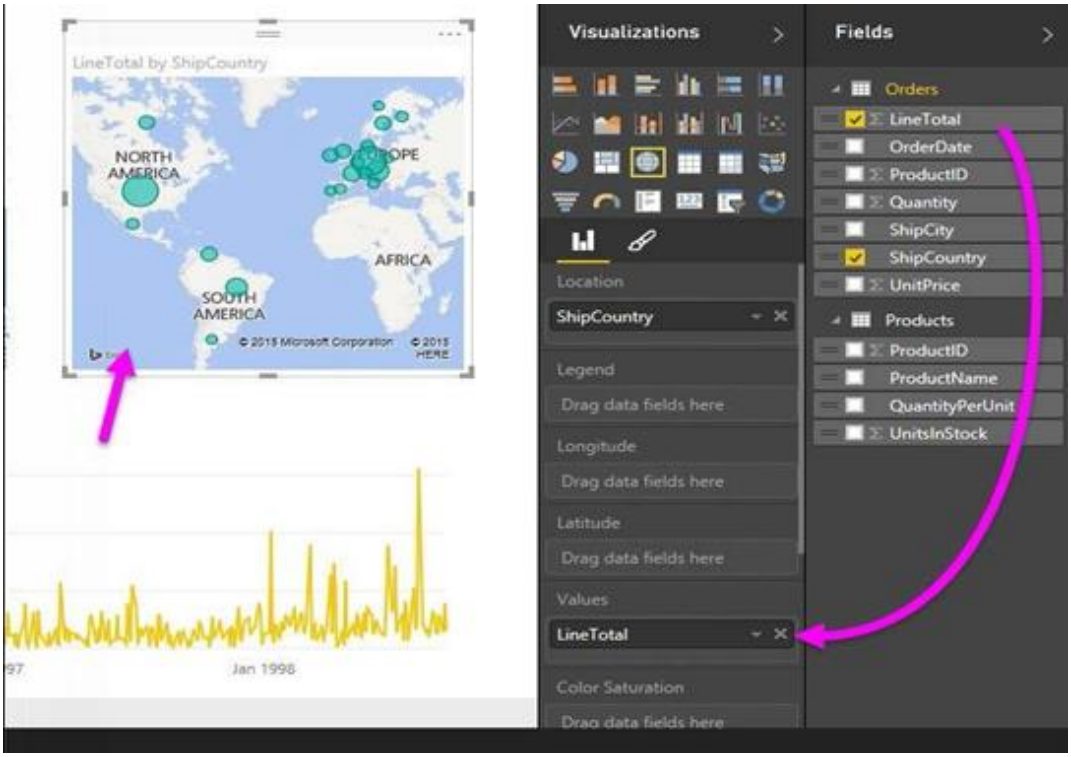
Task 4: Build visuals using your data

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





2. 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 2: Interact with your report visuals to analyze further