INDEX

		4. Fill All Missing Values with the Mean, Median, Mode, Minimum, and Maximum of the Particular Numeric Column			
	B	Write python/R program to create the network routing diagram from the given data on routers in assess superstep.		11/11/2022	
	C	Write python/R program to build directed acyclic graph. Example 1: Company location DAG Example 2: Customer location DAG Example 3: DAG using GPS data		12/11/2022	
	D	Write python/R program to pick the content for Billboards for the given data.		12/11/2022	
5		**Processing Data**			
	A	Build the time hub, links and satellites.		18/11/2022	
	B	Golden Nominal		18/11/2022	
	C	Vehicles as object hub and satellite using python and SQLite.		19/11/2022	
	D	Human-Environment interaction		19/11/2022	
6		**Transforming Data**			
	A	Program to show the details of hub: person being born.		25/11/2022	
	B	Building dimension Person, Time and fact PersonBornAtTime		25/11/2022	
	C	Building a Data warehouse for transform superstep.		25/11/2022	
	D	Write python program to demonstrate Simple Linear Regression.		25/11/2022	
7		**Organizing Data**			
	A	Write python/R program to perform the horizontal style subset or slice of the data warehouse data.		26/11/2022	
	B	Write python/R program to perform the vertical style subset or slice of the data warehouse data.		26/11/2022	
	C	Write python/R program to perform the island style subset or slice of the data warehouse data.		26/11/2022	
	D	Write python/R program to perform the secure vault style subset or slice of the data warehouse data and attach the result to the person who performs the query.		26/11/2022	
	E	Write python program to demonstrate Association rule mining.		26/11/2022	
	F	Write python program to create network routing diagram in organizing superstep.		26/11/2022	
8		**Generating Data**			
	A	Write python/R program to perform data visualization to create following graphs using profit data. 1. Pie Graph 2. Double Pie Graph		2/12/2022	

		3. Line Graph 4. Vertical Bar Graph 5. Horizontal Bar Graph 6. Area Graph 7. Scatter Graph 8. Hex Bin Graph			
	B	Write python/R program to perform data visualization to create following advanced graphs/plots for the data. 1. Kernel Density Estimation (KDE) Graph 2. Scatter Matrix 3. Andrew's Curves 4. Parallel Coordinates 5. RADVIZ method 6. Lag Plot 7. Autocorrelation Plot 8. Bootstrap Plot 9. Contour Graphs 10. 3D Graph		2/12/2022	
9		**Data Visualization with Power BI.**		3/12/2022	

Practical 1

AIM: Write Python program to convert files from different formats to HORUS format .

Theory:

The Homogeneous Ontology for Recursive Uniform Schema (HORUS) is used as an internal data format structure that enables the framework to reduce the permutations of transformations required by the framework. The use of HORUS methodology results in a hub-and-spoke data transformation approach. External data formats are converted to HORUS format, and then a HORUS format is transformed into any other external format. The basic concept is to take native raw data and then transform it first to a single format. That means that there is only one format for text files, one format for JSON or XML, one format for images and video. Therefore, to achieve any-to-any transformation coverage, the framework's only requirements are a dataformat- to-HORUS and HORUS-to- data-format converter.

A.Conversion from Text Delimited CSV format to HORUS format.

Code:

```
#Utility Start CSV to Horus
#standard tools
#=========================================================
import pandas as pd
#input agreement=======================================
sInputFileName='D:/MEL/MSC-IT/Part 1/SEM 1/Data
Science/Practicals/2A/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('==========Melissa 53004220035================')
#Output Agreement========================================
OutputData=ProcessData
sOutputFileName='D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/Practicals/2A/HORUS-CSV-
Country.csv'
OutputData.to_csv(sOutputFileName,index=False)
print('CSV to HORUS - Done')
#Utility done
```

Output:

```
In [1]: runfile('C:/Users/MELISSA A DSOUZA/.spyder-py3/temp.py', wdir='C:/Users/
MELISSA A DSOUZA/.spyder-py3')
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]
==========Melissa 53004220035================
CSV to HORUS - Done
```

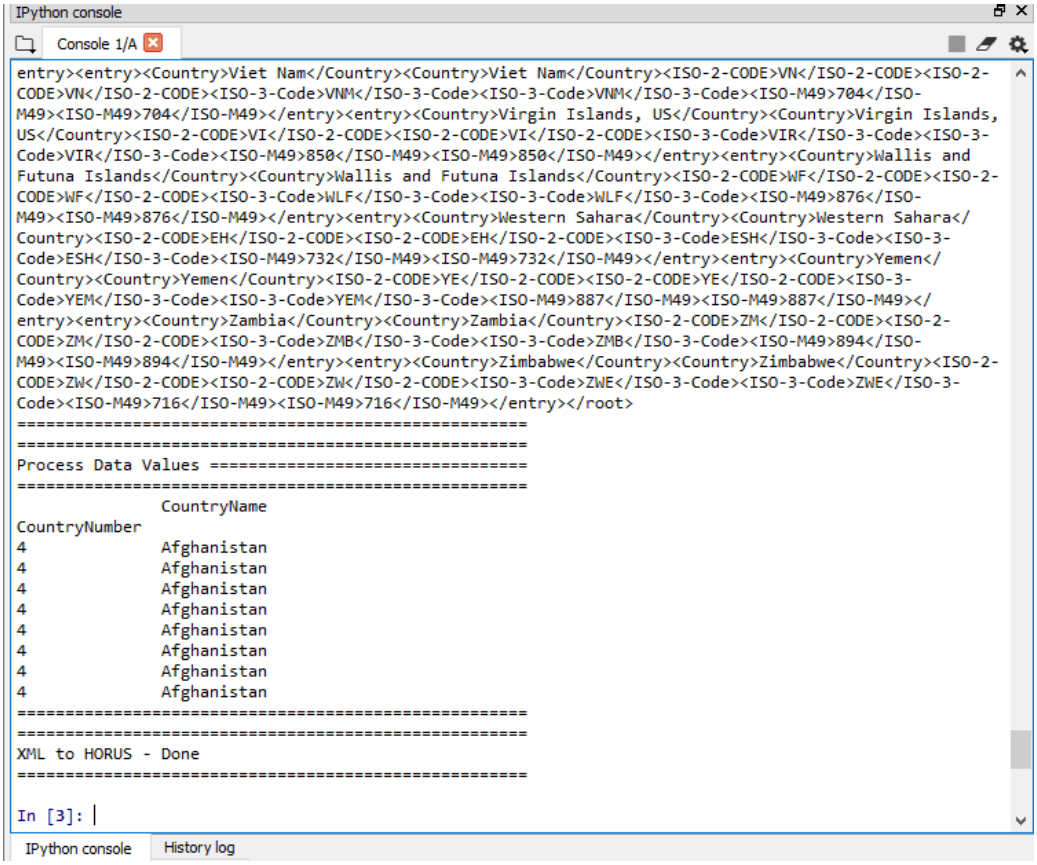
B. Conversion from XML format to HORUS format.

Code:

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

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

InputData = open(sInputFileName).read()

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

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

```
#============================================================
ProcessDataXML=InputData

# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)

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

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)

# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)

print('================================================')
print('Process Data Values ============================')
print('================================================')
print(ProcessData)
print('================================================')
#============================================================
# Output Agreement ==========================================
#============================================================
OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)

print('================================================')
print('XML to HORUS - Done')
print('================================================')
# Utility done ==============================================
```

Output:



C. Conversion from JSON to HORUS format.

Code:

```
Import pandas as pd
#Input Agreement
sInputFileName='D:/MEL/MSC-IT/Part 1/SEM 1/Data
Science/Practicals/1C/Country_Code.json'
InputData=pd.read_json(sInputFileName,orient='index',encoding="latin-1")
print('Input Data Values')
print(InputData)
print('========Melissa53004220035==========')

#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 currency number
ProcessData.sort_values('CountryName',axis=0,ascending=False,inplace=True)
print('Process Data Values')
print(ProcessData)

#Output Agreement
OutputData=ProcessData
sOutputFileName='D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/Practicals/1C/HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName,index=False)
print('JSON to HORUS-Done')
```

Output:

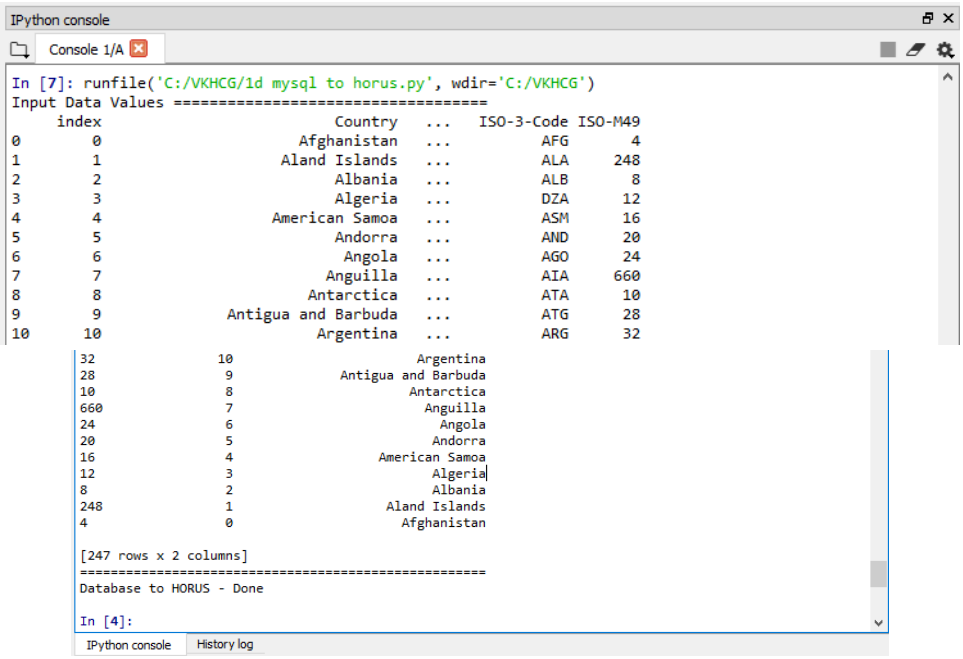
```
In [2]: runfile('C:/Users/MELISSA A DSOUZA/untitled0.py', wdir='C:/Users/MELISSA A DSOUZA')
Input Data Values
                       Country ISO-2-CODE ISO-3-Code  ISO-M49
0                  Afghanistan          AF         AFG        4
1                Aland Islands          AX         ALA      248
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]
========Melissa53004220035==========
========Melissa53004220035==========
Process Data Values
                                CountryName
CountryNumber
716                               Zimbabwe
894                                 Zambia
887                                  Yemen
732                         Western Sahara
876             Wallis and Futuna Islands
...                                    ...
16                          American Samoa
12                                 Algeria
8                                  Albania
248                          Aland Islands
4                              Afghanistan

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

D. Conversion from MYSQL Database to HORUS format.

Code:

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

print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
# Processing Rules ========================================
ProcessData=InputData

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

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)

# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)

print('Process Data Values =================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ========================================
OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)

print('Database to HORUS - Done')
# Utility done ============================================
```

Output:



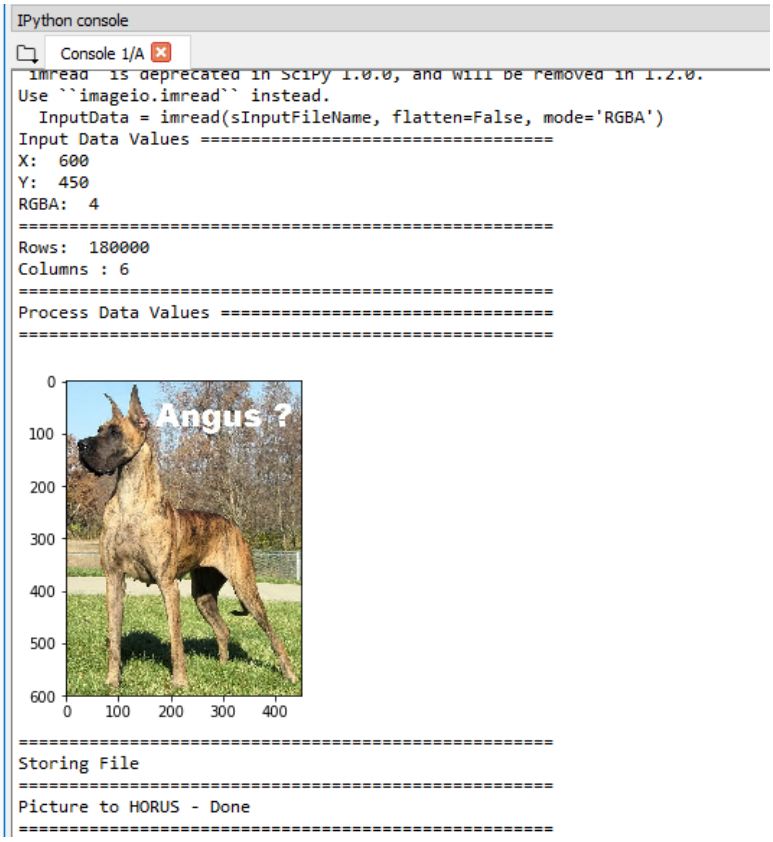
E. Conversion from Picture (JPEG) to HORUS format.

Code:

```
# Utility Start Picture to HORUS ================================
# Standard Tools
#================================================================
from scipy.misc import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Input Agreement ========================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/Angus.jpg'
InputData = imread(sInputFileName, flatten=False, mode='RGBA')

print('Input Data Values ===================================')
print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])
print('=====================================================')
# Processing Rules ========================================
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']
ProcessData.columns=sColumns
```

```
ProcessData.index.names =['ID']
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
print('=====================================================')
print('Process Data Values ================================')
print('=====================================================')
plt.imshow(InputData)
plt.show()
print('=====================================================')
# Output Agreement ========================================
OutputData=ProcessData
print('Storing File')
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Picture to HORUS - Done')
print('=====================================================')
# Utility done ============================================
```

Output:



F. Conversion from Video to HORUS format.

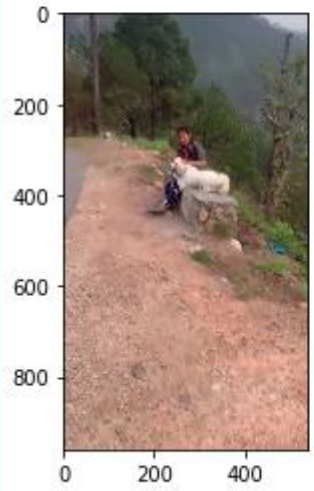
Code:

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

Output:

```
============= Input Data Values =============
X:  960
Y:  540
RGBA:  4
=====================================================
=====================================================
Process Data Values ================================
=====================================================
```

```
============= Input Data Values =============
X:  960
Y:  540
RGBA:  4
=====================================================
=====================================================
Process Data Values ================================
=====================================================
```




```
-------------------------------------
Movie to Frame-> Frame to Horus - Done by
```

G. Conversion from Audio to HORUS format.

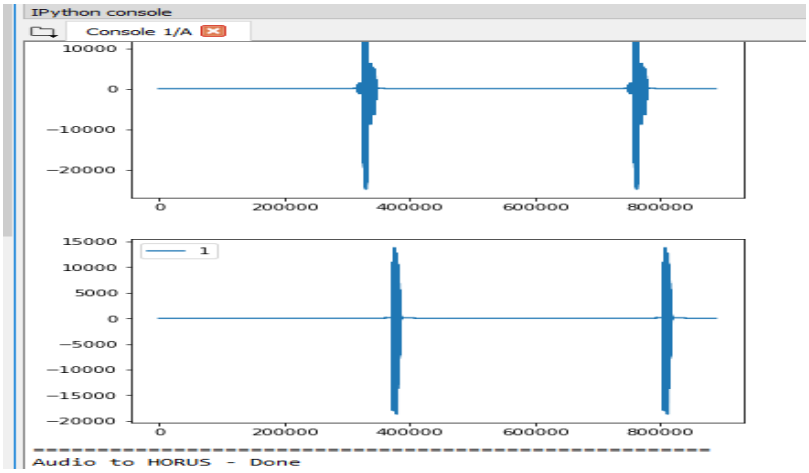
Code:

```
# Utility Start Audio to HORUS ============================
# Standard Tools
#==========================================================
from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#==========================================================
def show_info(aname, a,r):
    print ('---------------')
    print ("Audio:", aname)
```

```
    print ('---------------')
    print ("Rate:", r)
    print ('---------------')
    print ("shape:", a.shape)
    print ("dtype:", a.dtype)
    print ("min, max:", a.min(), a.max())
    print ('---------------')
    plot_info(aname, a,r)
#============================================================
def plot_info(aname, a,r):
    sTitle= 'Signal Wave - '+ aname + ' at ' + str® + 'hz'
    plt.title(sTitle)
    sLegend=[]
    for c in range(a.shape[1]):
        sLabel = 'Ch' + str(c+1)
        sLegend=sLegend+[str(c+1)]
        plt.plot(a[:,c], label=sLabel)
    plt.legend(sLegend)
    plt.show()
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("4 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
```

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

Output:



Practical 2

AIM: Utilities and Auditing

Theory:

Basic Utility Design

1. Load data as per input agreement.
2. Apply processing rules of utility.
3. Save data as per output agreement.

There are three types of utilities
- Data processing utilities
- Maintenance utilities
- Processing utility

A.Fixer Utility.

1. Removing leading and lagging spaces from data entry.
2. Removing non-printable characters from data entry.
3. Reformatting data entry to match specific formatting criteria.

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(2022,11,10)
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:

```
In [1]: runfile('C:/Users/MELISSA A DSOUZA/untitled0.py', wdir='C:/Users/MELISSA A
DSOUZA')
1.Removing leading or lagging spaces from a data entry
> Data Science with too many spaces is bad!!! <
> Data Science with too many spaces is bad!!! <
2.Removing nonprintable characters from a data entry

Bad Data : Data Science with funny characters is bad!!!
Clean Data: join.(filter(lambda x: x in string.printable,baddata))
3.Reformatting data entry to match specific formatting criteria.
Bad Data: 2022-11-10
Good Data: 10November2022
```

B.Data Binning or Bucketing.

Binning is a data preprocessing technique used to reduce the effects of minor observation errors. Statistical data binning is a way to group a number of more or less continuous values into a smaller number of "bins."

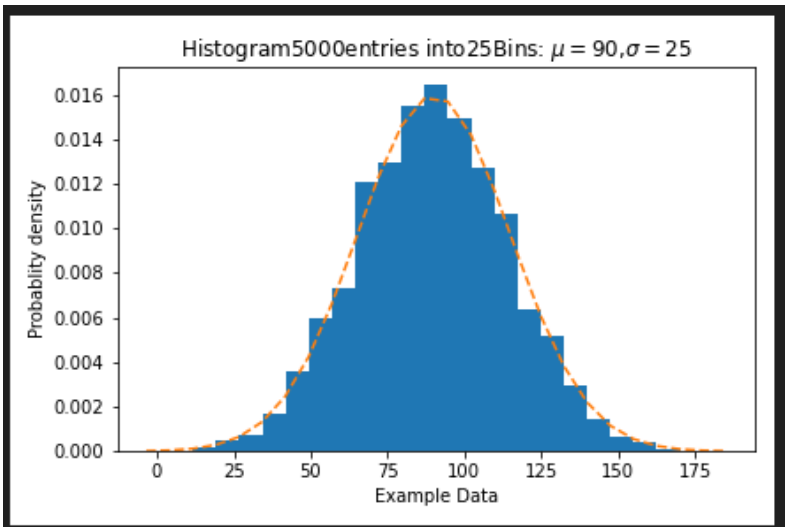
Code:

```
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('Probablity 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='D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/Practicals/2B/DU-Histogram.png'
fig.savefig(sPathFig)
plt.show()
```

Output:

```
In [5]: runfile('D:/MEL/MSC-IT/Part 1/SEM 1/Data
Science/Practicals/2B/DS_Prac2B.py', wdir='D:/MEL/MSC-
IT/Part 1/SEM 1/Data Science/Practicals/2B')
```



C. Averaging of Data.

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

Code:

```
import pandas as pd

InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/Practicals/2C/'
```

```
print('========Melissa53004220035========')
print('Working Base:',Base,'using')
print('==================================')
sFileName = Base + InputFileName
print('Loading:',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,usecols=['Country','P
lace 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:

```
In [4]: runfile('D:/MEL/MSC-IT/Part 1/SEM 1/Data
Science/Practicals/2C/DS_Prac2C.py', wdir='D:/
MEL/MSC-IT/Part 1/SEM 1/Data Science/Practicals/
2C')
========Melissa53004220035========
Working Base: D:/MEL/MSC-IT/Part 1/SEM 1/Data
Science/Practicals/2C/ using
==================================
Loading: D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/
Practicals/2C/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

In [5]:
```

D. Outlier Detection.

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

Code:

```
import pandas as pd

InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/Practicals/2D/'
print('========Melissa53004220035========')
print('Working Base:',Base)
print('================================')
sFileName = Base + InputFileName
print('Loading:',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,usecols=['Country','P
lace 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('AllData')
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:

```
========Melissa53004220035========
Working Base: D:/MEL/MSC-IT/Part 1/SEM 1/Data
Science/Practicals/2D/
==================================
Loading: D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/
Practicals/2D/IP_DATA_CORE.csv
AllData
     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 51.512635507867415
     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 51.506176875621264
     Country Place_Name  Latitude
1915      GB     London   51.4739
Not Outliers
     Country Place_Name  Latitude
1917      GB     London   51.5085
1918      GB     London   51.5085
1922      GB     London   51.5085
1928      GB     London   51.5085
1929      GB     London   51.5085
...      ...        ...       ...
3432      GB     London   51.5092
3433      GB     London   51.5092
3434      GB     London   51.5092
```

E. Logging. Write Python program for basic logging in Data Science.

Code:

```
import pandas as pd
import sys
import os
import logging
import uuid
import shutil
import time
#############################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
#############################################################
sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04-Clark']
sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-Organise','06-
Report']
sLevels=['debug','info','warning','error']

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

```
        # set a format which is simpler for console use
        formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
        # tell the handler to use this format
        console.setFormatter(formatter)
        # add the handler to the root logger
        logging.getLogger('').addHandler(console)

        # Now, we can log to the root logger, or any other logger. First the
root...
        logging.info('Practical Data Science is fun!.')

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

            if sLevel == 'debug':
                logger.debug('Practical Data Science logged a debugging
message.')

            if sLevel == 'info':
                logger.info('Practical Data Science logged information message.')

            if sLevel == 'warning':
                logger.warning('Practical Data Science logged a warning
message.')

            if sLevel == 'error':
                logger.error('Practical Data Science logged an error message.')
```

Output:

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Durgadatt/Desktop/file/Pracs ds/3e.py', wdir='C:/Users/Durgadatt/
Desktop/file/Pracs ds')
root         : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-01-Retrieve-info: INFO     Practical Data Science logged information
message.
Apllication-01-Vermeulen-01-Retrieve-warning: WARNING  Practical Data Science logged a warning
message.
Apllication-01-Vermeulen-01-Retrieve-error: ERROR    Practical Data Science logged an error
message.
Set up: C:/VKHCG/01-Vermeulen/01-Retrieve/Logging/Logging_a5170d39-ce78-414b-
bb6c-1aa77b6722af.log
root         : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-02-Assess-info: INFO     Practical Data Science logged information
message.
```

Practical 3

AIM: Retrieve Superstep

Theory:

The Retrieve superstep is a practical method for importing completely into the processing ecosystem a data lake consisting of various external data sources. The Retrieve superstep is the first contact between your data science and the source systems. I will guide you through a methodology of how to handle this discovery of the data up to the point you have all the data you need to evaluate the system you are working with, by deploying your data science skills. The successful retrieval of the data is a major stepping-stone to ensuring that you are performing good data science. Data lineage delivers the audit trail of the data elements at the lowest granular level, to ensure full data governance.

A. Program the following data processing using R.

Code:

```
 library(readr)
Warning message:
package 'readr' was built under R version 4.1.3
> IP_DATA_ALL<-read.csv(C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv)
Error: unexpected '/' in "IP_DATA_ALL<-read.csv(C:/"
> IP_DATA_ALL<-read.csv("C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
> View(IP_DATA_ALL)
```

Data: IP_DATA_ALL

	X	ID	Country	Place.Name
1	1	1	BW	Gaborone
2	2	2	BW	Gaborone
3	3	3	BW	Gaborone
4	4	4	BW	Gaborone
5	5	5	BW	Gaborone
6	6	6	BW	Gaborone
7	7	7	BW	Gaborone
8	8	8	BW	Gaborone
9	9	9	BW	Gaborone
10	10	10	BW	Gaborone

```
>spec(IP_DATA_ALL)
>library(tibble)
>set_tidy_names(IP_DATA_ALL,syntactic=TRUE,quiet=FALSE)
>IP_DATA_ALL_FIX=set tidy names(IP_DATA_ALL, syntactic= TRUE, quiet=TRUE)
>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['Co
untry'])==0,]$Country))
>setorder(hist_country,'Country')
>hist_country_with_id=rowid_to_column(hist_country,var="RowIDCountry")
>View(hist_country_fix)
>IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_FIX,table(Country)))
>View(IP_DATA_COUNTRY_FREQ)
```

	Country
1	AD
2	AE
3	AF
4	AG
5	AI
6	AL
7	AM
8	AO
9	AR
10	AS

```
>sapply(IP_DATA_ALL_FIX[,'Latitude'],min,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Country'],min,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Latitude'],max,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Country'],max,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Latitude'],mean,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Latitude'],median,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Latitude'],range,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Latitude'],quantile,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Latitude'],sd,na.rm=TRUE)
>sapply(IP_DATA_ALL_FIX[,'Longitude'],sd,na.rm=TRUE)
```

Output:

	Country	N
1	AD	46
2	AE	1793
3	AF	15
4	AG	21
5	AI	9
6	AL	91
7	AM	92
8	AO	108
9	AR	120
10	AS	5

B. Program to retrieve different attributes of data.

Code:

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
################################################################
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:

```
In [1]: runfile('D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/
Practicals/2F/untitled0.py', wdir='D:/MEL/MSC-IT/Part 1/SEM 1/
Data Science/Practicals/2F')
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 1247502
Columns: 9
### Raw Data Set #####################################
Unnamed: 0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
### Fixed Data Set ###################################
Unnamed:.0 <class 'str'>
```

```
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
### Done!! ############################################
```

C. Data Pattern using R.

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

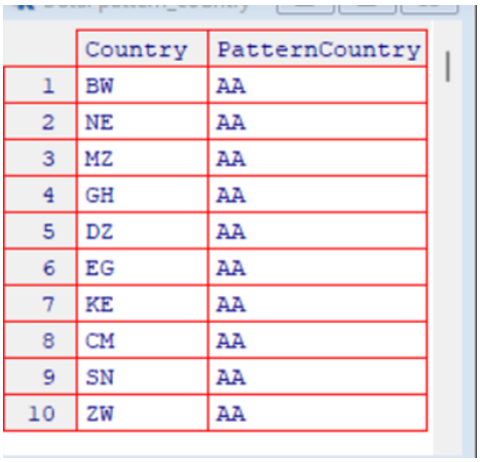
Code:

```
> library(readr)
> library(data.table)
data.table 1.14.4 using 2 threads (see ?getDTthreads).  Latest news: r-
datatable.com
> IP_DATA_ALL<-read_csv(FileName)
Error in standardise_path(file) : object 'FileName' not found
> FileName=paste0('C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv')
> IP_DATA_ALL<-read_csv(FileName)
```

```
[1mindexing[0m [34mIP_DATA_ALL.csv[0m [==------------------------------]
[32m2.15GB/s[0m, eta: [36m 0s[0m
[1mindexing[0m [34mIP_DATA_ALL.csv[0m [==============================]
[32m497.14MB/s[0m, eta: [36m 0s[0m

New names:
• `` -> `...1`
Rows: 1247502 Columns: 9
── Column specification ────────────────────────────────────────────────────
Delimiter: ","
chr (3): Country, Place.Name, Post.Code
dbl (6): ...1, ID, Latitude, Longitude, First.IP.Number, Last.IP.Number

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
>
> hist_country=data.table(Country=unique(IP_DATA_ALL$Country))
>
pattern_country=data.table(Country=hist_country$Country,PatternCountry=hist_count
ry$Country)
> oldchar=c(letters,LETTERS)
> newchar=replicate(length(oldchar),"A")
>

> for(r in seq(nrow(pattern_country)))>
> 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>
> 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)
> library(readr)
> library(data.table)
> Base='C:/VKHCG'
> FileName=paste0('C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv')
```

```
> IP_DATA_ALL<-read_csv(FileName)

[1mindexing[0m [34mIP_DATA_ALL.csv[0m [=--------------------------------]
[32m2.15GB/s[0m, eta: [36m 0s[0m
[1mindexing[0m [34mIP_DATA_ALL.csv[0m [================================]
[32m459.20MB/s[0m, eta: [36m 0s[0m
[1mindexing[0m [34mIP_DATA_ALL.csv[0m [================================]
[32m456.06MB/s[0m, eta: [36m 0s[0m

New names:
• `` -> `...1`
Rows: 1247502 Columns: 9
── Column specification ────────────────────────────────────────────────────
Delimiter: ","
chr (3): Country, Place.Name, Post.Code
dbl (6): ...1, ID, Latitude, Longitude, First.IP.Number, Last.IP.Number

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> hist_latitude=data.table(Latitude=unique(IP_DATA_ALL$Latitude))
> pattern_latitude=data.table(latitude=hist_latitude$Latitude,
+ Patternlatitude=as.character(hist_latitude$Latitude))
> oldchar=c(letters,LETTERS)
> oldc rah = (c le tt ,sre L )

> newchar=replicate(length(oldchar),"A")
> for(r in seq(nrow(pattern_latitude))){
+ s=pattern_latitude[r,]$Patternlatitude;
+ for(c in seq(length(oldchar))){
+ s=chartr(oldchar[c],newchar[c],s)
+ };
+ for (n in seq(0,9,1)
+ ){
+ s=chartr(as.character(n),"N",s)
+ };
+ s=chartr("","b",s)
+ s=chartr("+","u",s)
+ s=chartr("-","u",s)
+ s=chartr(".","u",s)
+ pattern_latitude[r,]$Patternlatitude=s;
+ };
> setorder(pattern_latitude,latitude)
> View(pattern_latitude[1:3])
```

Output:



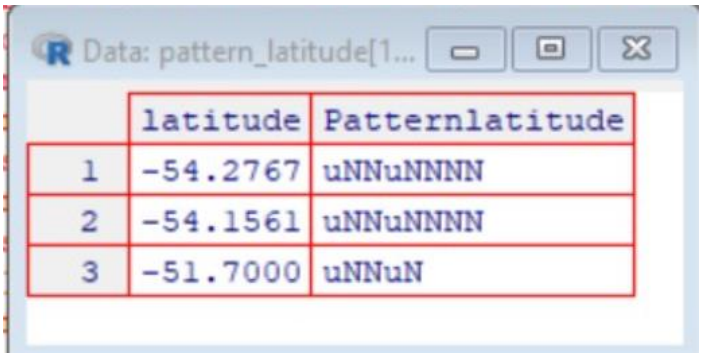
	Country	PatternCountry
1	BW	AA
2	NE	AA
3	MZ	AA
4	GH	AA
5	DZ	AA
6	EG	AA
7	KE	AA
8	CM	AA
9	SN	AA
10	ZW	AA

Example 2

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

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

Output:



	latitude	Patternlatitude
1	-54.2767	uNNuNNNN
2	-54.1561	uNNuNNNN
3	-51.7000	uNNuN

D.

1] Loading IP_DATA_ALL.

This data set contains all the IP address allocation in the world. It will help you to locate your customer when interacting with them online.

Create a new python script and save it as Retrieve-IP_DATA_ALL.py in directory.

Code:

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
################################################################
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:

```
In [1]: runfile('D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/
Practicals/2F/untitled0.py', wdir='D:/MEL/MSC-IT/Part 1/SEM 1/
Data Science/Practicals/2F')
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 1247502
Columns: 9
### Raw Data Set #####################################
Unnamed: 0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
### Fixed Data Set ##################################
Unnamed:.0 <class 'str'>
```

```
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
### Done!! ############################################
```

2] Haversine distance calculating using Vermeulen dataset

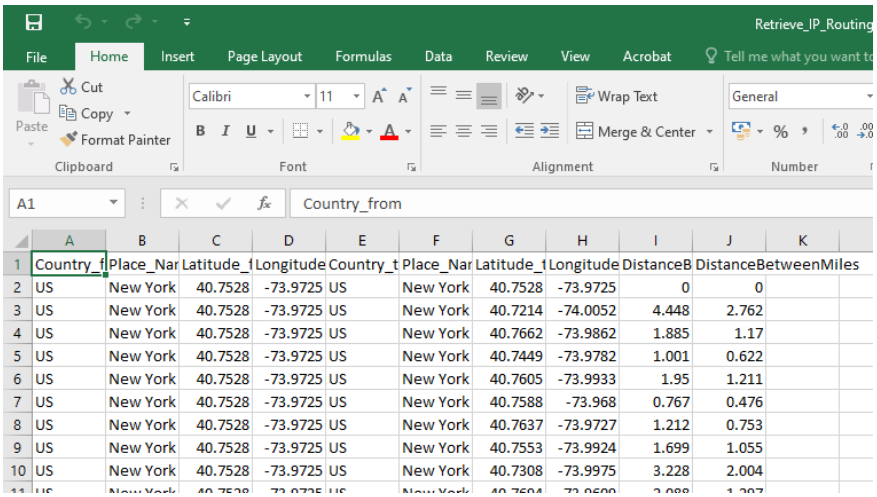
The company has two main jobs on which to focus your attention:
-> Designing a routing diagram for company
->Planning a schedule of jobs to be performed for the router network.

Code:

```
#################################################################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
from math import radians, cos, sin, asin, sqrt
#################################################################
def haversine(lon1, lat1, lon2, lat2,stype):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    if stype == 'km':
        r = 6371 # Radius of earth in kilometers
    else:
        r = 3956 # Radius of earth in miles
    d=round(c * r,3)
    return d
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
#################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_CORE.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
#################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
IP_DATA = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
```

```
IP_DATA.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA1 = IP_DATA
IP_DATA1.insert(0, 'K', 1)
IP_DATA2 = IP_DATA1
#################################################################
print(IP_DATA1.shape)
#################################################################
IP_CROSS=pd.merge(right=IP_DATA1,left=IP_DATA2,on='K')
IP_CROSS.drop('K', axis=1, inplace=True)
IP_CROSS.rename(columns={'Longitude_x': 'Longitude_from', 'Longitude_y':
'Longitude_to'}, inplace=True)
IP_CROSS.rename(columns={'Latitude_x': 'Latitude_from', 'Latitude_y':
'Latitude_to'}, inplace=True)
IP_CROSS.rename(columns={'Place_Name_x': 'Place_Name_from', 'Place_Name_y':
'Place_Name_to'}, inplace=True)
IP_CROSS.rename(columns={'Country_x': 'Country_from', 'Country_y': 'Country_to'},
inplace=True)
#################################################################
IP_CROSS['DistanceBetweenKilometers'] = IP_CROSS.apply(lambda row:
    haversine(
            row['Longitude_from'],
            row['Latitude_from'],
            row['Longitude_to'],
            row['Latitude_to'],
            'km')
            ,axis=1)
#################################################################
IP_CROSS['DistanceBetweenMiles'] = IP_CROSS.apply(lambda row:
    haversine(
            row['Longitude_from'],
            row['Latitude_from'],
            row['Longitude_to'],
            row['Latitude_to'],
            'miles')
            ,axis=1)
print(IP_CROSS.shape)
sFileName2=sFileDir + '/Retrieve_IP_Routing.csv'
IP_CROSS.to_csv(sFileName2, index = False, encoding="latin-1")
#################################################################
print('### Done!! ############################################')
#################################################################
```

Output:



	A	B	C	D	E	F	G	H	I	J
1	Country_f	Place_Nar	Latitude_f	Longitude	Country_t	Place_Nar	Latitude_t	Longitude	DistanceB	DistanceBetweenMiles
2	US	New York	40.7528	-73.9725	US	New York	40.7528	-73.9725	0	0
3	US	New York	40.7528	-73.9725	US	New York	40.7214	-74.0052	4.448	2.762
4	US	New York	40.7528	-73.9725	US	New York	40.7662	-73.9862	1.885	1.17
5	US	New York	40.7528	-73.9725	US	New York	40.7449	-73.9782	1.001	0.622
6	US	New York	40.7528	-73.9725	US	New York	40.7605	-73.9933	1.95	1.211
7	US	New York	40.7528	-73.9725	US	New York	40.7588	-73.968	0.767	0.476
8	US	New York	40.7528	-73.9725	US	New York	40.7637	-73.9727	1.212	0.753
9	US	New York	40.7528	-73.9725	US	New York	40.7553	-73.9924	1.699	1.055
10	US	New York	40.7528	-73.9725	US	New York	40.7308	-73.9975	3.228	2.004

E. Building a Diagram for the Scheduling of Jobs

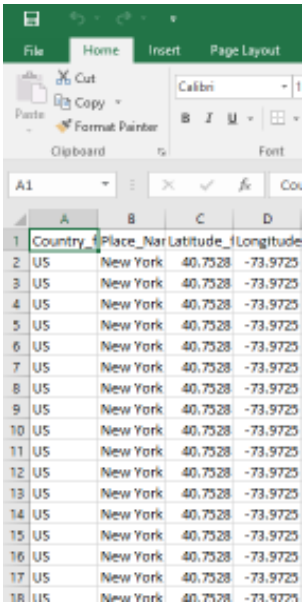
Start your Python editor and create a text file named Retrieve-Router-Location,py in directory

Code:

```
#################################################################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
#################################################################
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
#################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
#################################################################
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
#################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
```

```
print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])

sFileName2=sFileDir + '/' + OutputFileName
ROUTERLOC.to_csv(sFileName2, index = False, encoding="latin-1")
#################################################################
print('### Done!! ############################################')
#################################################################
```

Output:



F. Picking Content for billboards

Start your Python editor and create a text file named Retrieve-DE-Billboard-Location,py in directory

Code:

```
#################################################################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
#################################################################
InputFileName='DE_Billboard_Locations.csv'
OutputFileName='Retrieve_DE_Billboard_Locations.csv'
Company='02-Krennwallner'
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
```

```
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Base='C:/VKHCG'
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','PlaceName','Latitude','Longitude'])

IP_DATA_ALL.rename(columns={'PlaceName': 'Place_Name'}, inplace=True)
################################################################
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)

print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])

sFileName2=sFileDir + '/' + OutputFileName
ROUTERLOC.to_csv(sFileName2, index = False)

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

Output:

```
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/02-Krennwallner/00-RawData/DE_Billboard_Locations.csv
Rows : 8873
Columns : 4
### Done!! ############################################

In [3]:
```

Practical 4

AIM: Assessing Data

Theory:

Data quality refers to the condition of a set of quantitative or qualitative variables. Dataquality is a multidimensional measurement of the acceptability of specific data sets. Inbusiness, data quality is measured to determine whether data can be used as a basis forreliable intelligence extraction for supporting organizational decisions. Data profiling involves observing in your data source all the viewpoints that the information offers. The main goal is to determine if individual viewpoints are accurate complete. The Assess superstep determines what additional processing to apply to the entries that are noncompliant.

Errors

Typically, one of four things can be done with an error to the data

1) Accept the Error
2) Reject the Error
3) Correct the Error
4) Create a Default Value

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

Missing Values in Pandas

1. Drop the Columns Where All Elements are Missing Values

Code:

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

```
#################################################################
Base='C:/VKHCG'
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
### Import Warehouse
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)

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

Output:

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

```
## 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
```

```
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!! ####################
################################
```

2. Drop the Columns Where any of the Elements is Missing Values

Code:

```
#################################################################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
#################################################################
Base='C:/VKHCG'
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-02.csv'
Company='01-Vermeulen'
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
#################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
### Import Warehouse
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)

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

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

Output:

```
In [8]: runfile('D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/
Practicals/untitled1.py', wdir='D:/MEL/MSC-IT/Part 1/SEM 1/Data
Science/Practicals')
##############################
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
```

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

```
13      14
14      15
15      16
16      17
17      18
18      19
19      20
20      21
################################
## Data Profile
################################
Rows : 21
Columns : 1
################################
################################
### Done!! ####################
################################
```

3. Keep only the rows that contain a maximum of two Missing Values

Code:

```
## -*- coding: utf-8 -*-
"""
Created on Sat Nov 26 12:57:10 2022

@author: admin
"""

###################### Assess-Good-Bad-03.py ################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
#################################################################
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-03.csv'
Company='01-Vermeulen'
Base='C:/VKHCG'
#################################################################
print('################################')
print('Working Base :',Base, ' using Windows ~~~~')
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
### Import Warehouse
#################################################################
```

```
sFileName=Base + '/' + Company + '/00-RawData/' + 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)
print('################################')
print('## Test Data Values')
print('################################')
print(TestData)
print('################################')
print('## Data Profile')
print('################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('################################')
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#################################################################
print('################################')
print('### Done!! ####################')
print('################################')
#################################################################
```

Output:

```
In [3]: runfile('C:/Users/admin/Desktop/pracs/DS/Practical 5Aiii.py',
wdir='C:/Users/admin/Desktop/pracs/DS')
################################
Working Base : C:/VKHCG  using Windows ~~~~
################################
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  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
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 : 20
Columns : 8
################################
################################
### Done!! ####################
```

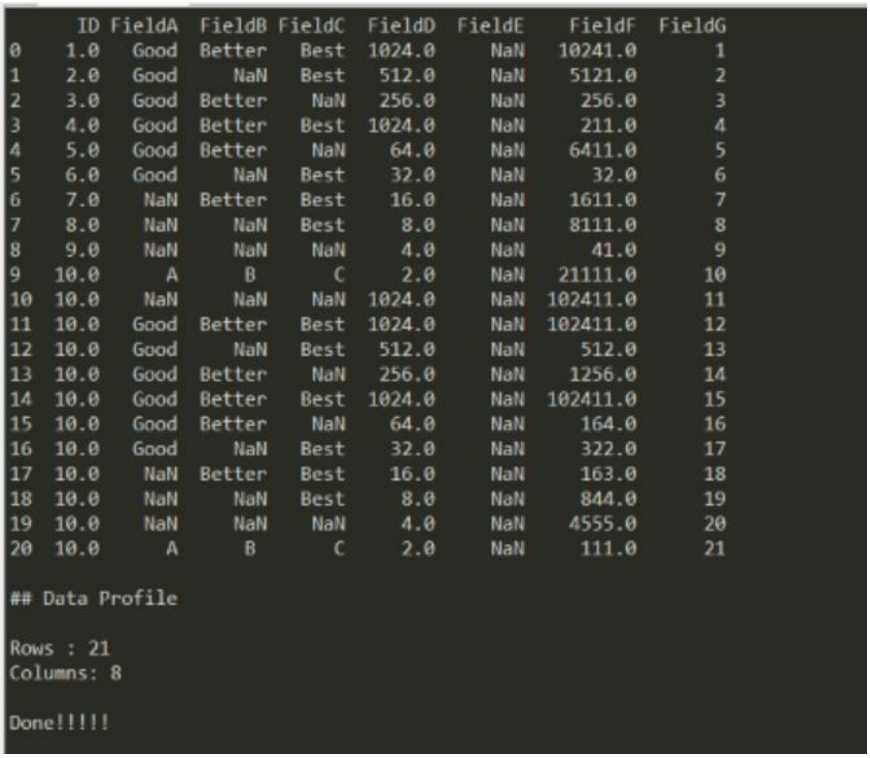
4. Fill all Missing Values with the Mean, Median, Mode, Minimum and Maximum of the Particular Numeric Column

Code:

```
#import sys
import os
import pandas as pd
Base='C:/VKHCG'
print('AIM: Fill all the missing values with Mean Median Mode Minimum Maximum of
numeric column\n  ')
print('Working Base :',Base, ' using ',sys.platform)
print('     ')
sInputFileName='Good-or-Bad.csv'
sOutputFileNameA='Good-or-Bad-04-A.csv' #MEAN
sOutputFileNameB='Good-or-Bad-04-B.csv' #MEDIAN
sOutputFileNameC='Good-or-Bad-04-C.csv' #MODE
sOutputFileNameD='Good-or-Bad-04-D.csv' #MIN
sOutputFileNameE='Good-or-Bad-04-E.csv' #MAX
Company='01-Vermeulen'
Base='C:/VKHCG'
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('     ')
print('Raw Data Values')
print(RawData)
print('     ')
print('Data Profile')
print('     ')
print('R-ows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
TestData=RawData.fillna(RawData.mean())
print('     ')
print('## Test Data Values')
print(TestData)
print('     ')
print('## Data Profile')
print('     ')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
sFileName=sFileDir + '/' + sOutputFileNameA
TestData.to_csv(sFileName, index = False)
TestData=RawData.fillna(RawData.median())
print('     ')
print('## Test Data Values')
```

```
print(TestData)
print('     ')
print('## Data Profile')
print('     ')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
sFileName=sFileDir + '/' + sOutputFileNameB
TestData.to_csv(sFileName, index = False)
TestData=RawData.fillna(RawData.mode())
print('     ')
print('## Test Data Values')
print(TestData)
print('     ')
print('## Data Profile')
print('     ')
print('Rows :',TestData.shape[0])
print('Columns:',TestData.shape[1])
print('     ')
sFileName=sFileDir + '/' + sOutputFileNameC
TestData.to_csv(sFileName, index = False)
TestData=RawData.fillna(RawData.min())
print('     ')
print('## Test Data Values')
print(TestData)
print('     ')
print('## Data Profile')
print('     ')
print('Rows :',TestData.shape[0])
print('Columns:',TestData.shape[1])
print('     ')
sFileName=sFileDir + '/' + sOutputFileNameD
TestData.to_csv(sFileName, index = False)
TestData=RawData.fillna(RawData.max())
print('     ')
print('## Test Data Values')
print(TestData)
print('     ')
print('## Data Profile')
print('     ')
print('Rows :',TestData.shape[0])
print('Columns:',TestData.shape[1])
print('     ')
sFileName=sFileDir + '/' + sOutputFileNameE
TestData.to_csv(sFileName, index = False)
print('Done!!!!!')
print('     ')
```

Output:



	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	1024.0	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	10.0	NaN	NaN	NaN	1024.0	NaN	102411.0	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	1024.0	NaN	102411.0	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

Done!!!!!
```

B. Write a Python program to create the network routing diagram from the given data on routers in assess superstep.

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('################################')
      ###################################################################
#################################################################
### Import Company Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('################################')
#################################################################
## Assess Company Data
#################################################################
print('################################')
print('Changed :',CompanyData.columns.values)
CompanyData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CompanyData.columns.values)
print('################################')
#################################################################
#################################################################
### Import Customer Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName3
print('################################')
print('Loading :',sFileName)
print('################################')
```

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

Output:

```
In [4]: runfile('C:/Users/admin/Desktop/pracs/DS/Practical 5Bi.py', wdir='C:/Users/admin/Desktop/
pracs/DS')
################################
Working Base : C:/VKHCG  using Windows
################################
################################
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv
################################
Loaded Country: ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
################################
################################
Changed : ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
To : ['Country_Name' 'Country_Code']
################################
################################
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv
################################
Loaded Company : ['Country' 'Place Name' 'Latitude' 'Longitude']
################################
################################
Changed : ['Country' 'Place Name' 'Latitude' 'Longitude']
To : ['Country_Code' 'Place Name' 'Latitude' 'Longitude']
################################
Change  ['Country_Code' 'Place Name' 'Latitude' 'Longitude' 'Country_Name']
To  ['Country_Code' 'Place Name' 'Latitude' 'Longitude' 'Company_Country_Name']
################################
################################
Storing : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv
################################
################################
### Done!! ####################
################################
```

C. Write a Python program to build Directed Cyclic Graph

Directed Acyclic Graph (DAG)
A directed acyclic graph is a specific graph that only has one path through the graph.

Example 1 : Company location DAG

Code:

```
################################################################
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
```

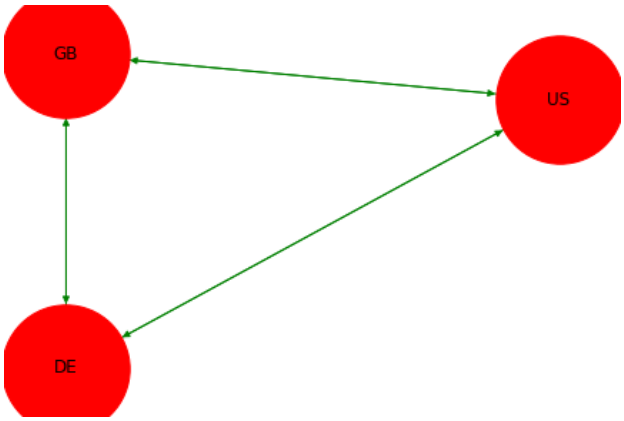
```
################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName1='Assess-DAG-Company-Country.png'
sOutputFileName2='Assess-DAG-Company-Country-Place.png'
Company='01-Vermeulen'
################################################################
### 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=Tr
ue,node_size=8000,font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
################################################################
print('################################')
for n1 in G2.nodes():
    for n2 in G2.nodes():
        if n1 != n2:
            print('Link :',n1,' to ', n2)
            G2.add_edge(n1,n2)
print('################################')

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

Output:

```
In [2]: runfile('D:/MEL/MSC-IT/Part 1/SEM 1/Data Science/
Practicals/2F/untitled0.py', wdir='D:/MEL/MSC-IT/Part 1/SEM 1/
Data Science/Practicals/2F')
################################
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
2        US   New York   40.7662   -73.9862
3        US   New York   40.7449   -73.9782
```

```
################################
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')]
################################
################################
Storing : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/
Assess-DAG-Company-Country.png
```



Example 2 : Customer location DAG

Code:

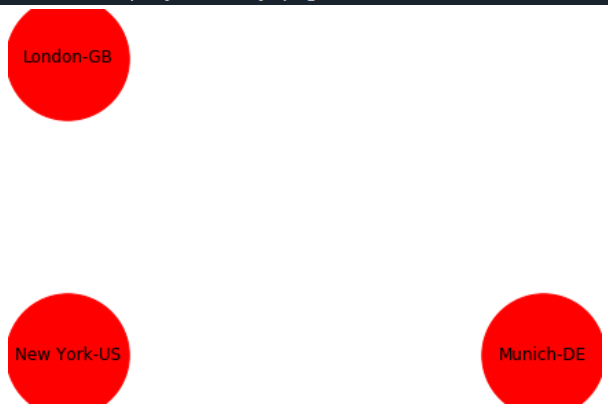
```
################################################################
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
sInputFileName='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=Tr
ue,node_size=8000,font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#################################################################
print('################################')
for n1 in G2.nodes():
    for n2 in G2.nodes():
        if n1 != n2:
            print('Link :',n1,' to ', n2)
            G2.add_edge(n1,n2)
print('################################')

print('################################')
print("Nodes of graph: ")
print(G2.nodes())
print("Edges of graph: ")
print(G2.edges())
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
sFileName=sFileDir + '/' + sOutputFileName2
print('################################')
print('Storing :', sFileName)
print('################################')
```

```
nx.draw(G2,pos=nx.spectral_layout(G2),nodecolor='r',edge_color='b',with_labels=Tr
ue,node_size=8000,font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
################################################################
```

Output:

```
##############################
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')]
##############################
##############################
Storing : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/
Assess-DAG-Company-Country.png
```



Example 3 : DAG using GPS data

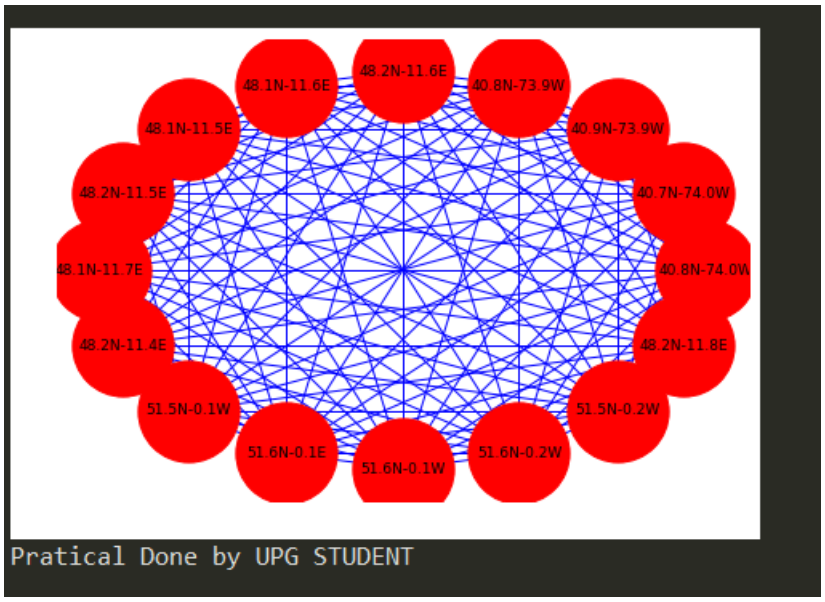
Code:

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

if sys.platform == 'windows':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
```

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

    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.nodes())
print("Edges of graph:")
print(G.edges())
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('################################')
pos=nx.circular_layout(G,dim=2, scale=2)
nx.draw(G,pos=pos,
        nodecolor='r',edge_color='b',
        with_labels=True,node_size=4000,
        font_size=9)
plt.savefig(sFileName) # save as png
plt.show()
print("Pratical Done by UPG STUDENT")
```

Output:



D. Write a Python program to pick the content for Billboards for the given data.

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('################################')
################################################################
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)
BillboardData.to_sql(sTable, conn, if_exists="replace")
print('#################')
################################################################
print(BillboardData.head())
```

```
print('################################')
print('Rows : ',BillboardData.shape[0])
print('################################')
#################################################################
### Import Billboard Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('################################')
print('Loading :',sFileName)
print('################################')
VisitorRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData[VisitorRawData.Country=='DE']
print('Loaded Company :',VisitorData.columns.values)
print('################################')
#################################################################
print('#################')
sTable='Assess_VisitorData'
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('#################')
sTable='Assess_BillboardVisitorData'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.Country AS BillboardCountry,"
sSQL=sSQL+ " A.Place_Name AS BillboardPlaceName,"
sSQL=sSQL+ " A.Latitude AS BillboardLatitude, "
sSQL=sSQL+ " A.Longitude AS BillboardLongitude,"
sSQL=sSQL+ " B.Country AS VisitorCountry,"
sSQL=sSQL+ " B.Place_Name AS VisitorPlaceName,"
sSQL=sSQL+ " B.Latitude AS VisitorLatitude, "
sSQL=sSQL+ " B.Longitude AS VisitorLongitude,"
sSQL=sSQL+ " (B.Last_IP_Number - B.First_IP_Number) * 365.25 * 24 * 12 AS
VisitorYearRate"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_BillboardData as A"
sSQL=sSQL+ " JOIN "
sSQL=sSQL+ " Assess_VisitorData as B"
sSQL=sSQL+ " ON "
sSQL=sSQL+ " A.Country = B.Country"
sSQL=sSQL+ " AND "
```

```
sSQL=sSQL+ " A.Place_Name = B.Place_Name;"
BillboardVistorsData=pd.read_sql_query(sSQL, conn)
print('#################')
#################################################################
print('#################')
sTable='Assess_BillboardVistorsData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardVistorsData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#################################################################
print(BillboardVistorsData.head())
print('################################')
print('Rows : ',BillboardVistorsData.shape[0])
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
print('################################')
print('Storing :', sFileName)
print('################################')
sFileName=sFileDir + '/' + sOutputFileName
BillboardVistorsData.to_csv(sFileName, index = False)
print('################################')
#################################################################
print('### Done!! ############################################')
#################################################################
```

Output:

```
Loading : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-
Python/Retrieve_DE_Billboard_Locations.csv
################################
Loaded Company : ['Country' 'Place_Name' 'Latitude'
'Longitude']
################################
#################
Storing : C:/VKHCG/02-Krennwallner/02-Assess/SQLite/
krennwallner.db  Table: Assess_BillboardData
#################
  Country         Place_Name  Latitude  Longitude
0      DE               Lake   51.7833     8.5667
1      DE               Horb   48.4333     8.6833
2      DE         Hardenberg   51.1000     7.7333
3      DE  Horn-bad Meinberg   51.9833     8.9667
4      DE             Winkel   51.5500    13.3833
################################
Rows :  8873
################################
```

Practical 5

AIM: Processing Data

Theory:

Hubs are the containers for business keys. They are the most important facets of the data vault methodology. The more successfully one is able to identify business keys the less refining of the model will follow.

Links stores the intersection of business keys (HUBS). Links can be considered the glue that holds the data vault model together. These tables allow for the data model to elegantly change over time because they can come and go as required by the business.

Satellites add all the color and description to the business keys (hubs) and relationships (links) in the data vault environment. Satellites contain all the descriptive information, tracking change by start and end dates over time, to let one know the information in effect at any point in time.

A. 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=1000
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:

```
2017-12-31 20:00:00
2017-12-31 19:00:00
2017-12-31 18:00:00
2017-12-31 17:00:00
2017-12-31 16:00:00
2017-12-31 15:00:00
2018-01-01 00:00:00
2017-12-31 23:00:00
2017-12-31 22:00:00
2017-12-31 21:00:00
2017-12-31 20:00:00
2017-12-31 19:00:00
2017-12-31 18:00:00
2017-12-31 17:00:00
2017-12-31 16:00:00
2017-12-31 15:00:00
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process-Time-Zulu
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Time-Zulu
################
Vacuum Databases
################
### Done!! ############################################
```

B. Golden Nominal.

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

Code:

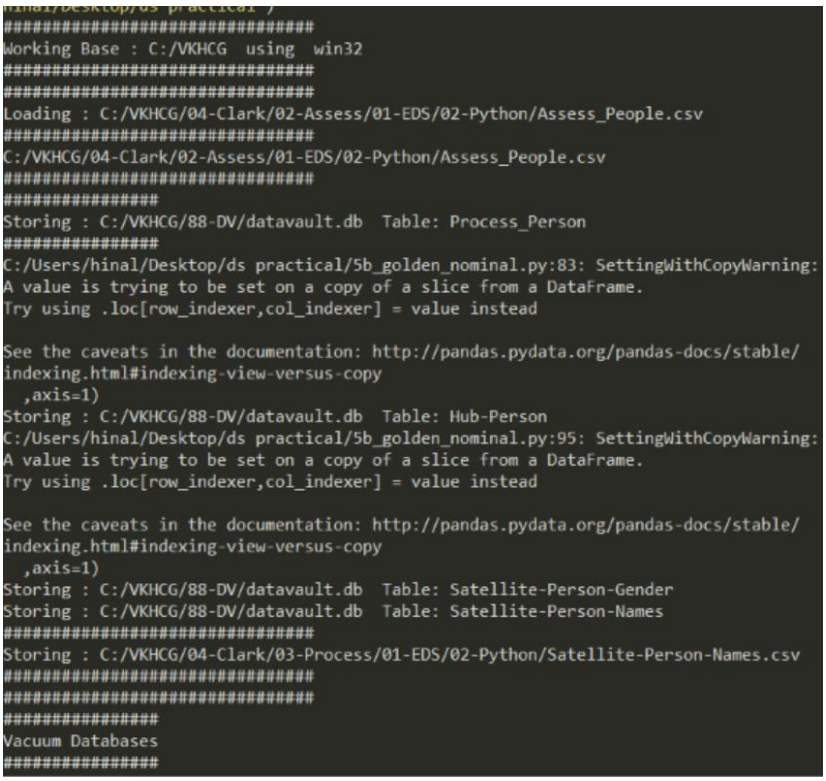
```
 #import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
from datetime import datetime, timedelta
from pytz import timezone, all_timezones
from random import randint
import uuid
################################################################
if sys.platform == 'linux':
    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:



C. Vehicles as object hub and satellite using python and SQLite.

Vehicles

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

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 == 'windows':
    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)
```

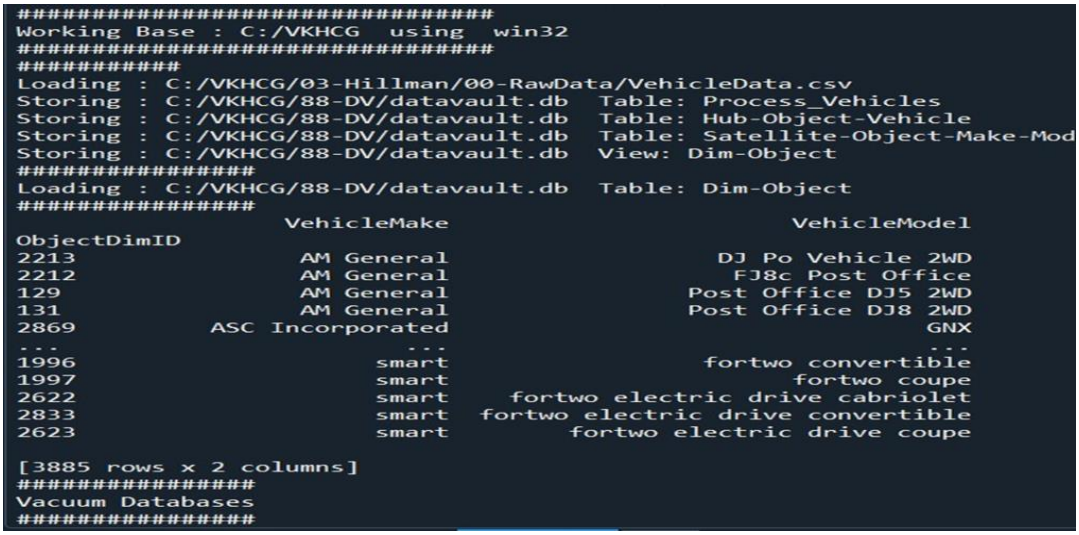
```
#############################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('###########')
print('Loading :',sFileName)
VehicleRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#############################################################
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:



D. Human- Environment interaction.

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

Code:

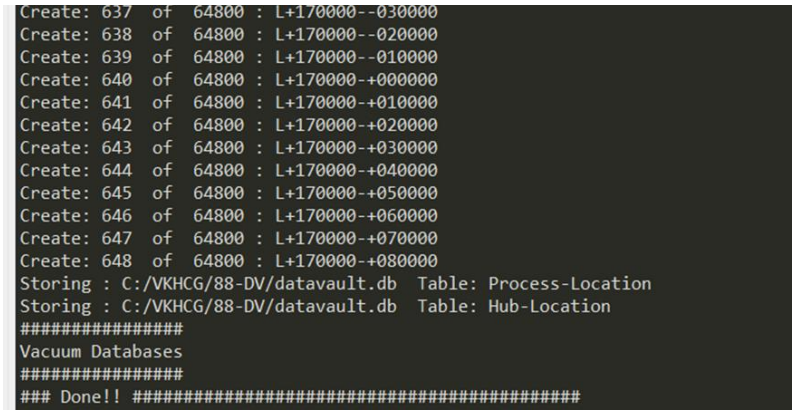
```
 ##################################################################

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

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

import sys

import os

import pandas as pd

import sqlite3 as sq

from pandas.io import sql

import uuid

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

Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

Company='01-Vermeulen'

InputAssessGraphName='Assess_All_Animals.gml'

EDSAssessDir='02-Assess/01-EDS'

InputAssessDir=EDSAssessDir + '/02-Python'

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

```
sFileAssessDir=Base + '/' + Company + '/' + InputAssessDir
if not os.path.exists(sFileAssessDir):
    os.makedirs(sFileAssessDir)
################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
t=0
tMax=360*180
################################################################
for Longitude in range(-180,180,10):
      for Latitude in range(-90,90,10):
      t+=1
        IDNumber=str(uuid.uuid4())
        LocationName='L'+format(round(Longitude,3)*1000, '+07d') +\
        '-'+format(round(Latitude,3)*1000, '+07d')
```

```
        print('Create:',t,' of ',tMax,':',LocationName)
        LocationLine=[('ObjectBaseKey', ['GPS']),
      ('IDNumber', [IDNumber]),
      ('LocationNumber', [str(t)]),
        ('LocationName', [LocationName]),
      ('Longitude', [Longitude]),
      ('Latitude', [Latitude])]
      if t==1:
            LocationFrame = pd.DataFrame.from_items(LocationLine)
      else:
                LocationRow = pd.DataFrame.from_items(LocationLine)
                LocationFrame = LocationFrame.append(LocationRow)
#################################################################
LocationHubIndex=LocationFrame.set_index(['IDNumber'],inplace=False)
#################################################################
sTable = 'Process-Location'
print('Storing :',sDatabaseName,' Table:',sTable)
LocationHubIndex.to_sql(sTable, conn1, if_exists="replace")
#################################################################
sTable = 'Hub-Location'
print('Storing :',sDatabaseName,' Table:',sTable)
LocationHubIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('#################')
print('Vacuum Databases')
sSQL="VACUUM;"
```

```
sql.execute(sSQL,conn1)

sql.execute(sSQL,conn2)

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

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

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

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

Output:



Practical 6
AIM: Transforming Data

Theory:

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

A. Program to show the details of hub: person being born.

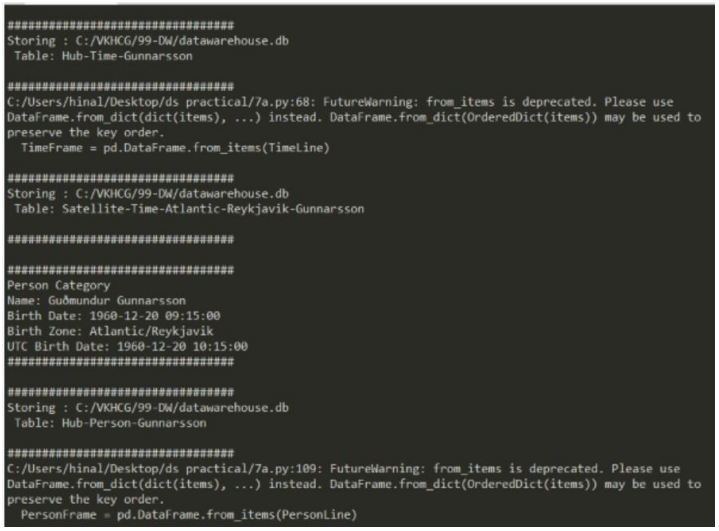
Code:

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
################################################################
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:



B. Building a dimension person, time and fact PersonBornAtTime.

Code:

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

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

```
PersonFrame = pd.DataFrame.from_items(PersonLine)
################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
################################################################
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
print('\n#################################')
print('Fact - Person - time')
print('\n#################################')
IDFactNumber=str(uuid.uuid4())
PersonTimeLine=[('IDNumber', [IDFactNumber]),
                ('IDPersonNumber', [IDPersonNumber]),
                ('IDTimeNumber', [IDTimeNumber])]
PersonTimeFrame = pd.DataFrame.from_items(PersonTimeLine)
################################################################
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
################################################################
sTable = 'Fact-Person-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
```

Output:

```
In [2]: runfile('G:/ds7b.py', wdir='G:')
################################
Working Base : C:/VKHCG  using  win32
################################

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

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

################################
G:/ds7b.py:59: FutureWarning: from_items is deprecated. Please use
DataFrame.from_dict(dict(items), ...) instead.
DataFrame.from_dict(OrderedDict(items)) may be used to preserve the key
order.
  TimeFrame = pd.DataFrame.from_items(TimeLine)

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

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

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

################################
G:/ds7b.py:83: FutureWarning: from_items is deprecated. Please use
DataFrame.from_dict(dict(items), ...) instead.
DataFrame.from_dict(OrderedDict(items)) may be used to preserve the key
order.
  PersonFrame = pd.DataFrame.from_items(PersonLine)

################################
Fact - Person - time

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

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

################################
G:/ds7b.py:102: FutureWarning: from_items is deprecated. Please use
DataFrame.from_dict(dict(items), ...) instead.
DataFrame.from_dict(OrderedDict(items)) may be used to preserve the key
order.
  PersonTimeFrame = pd.DataFrame.from_items(PersonTimeLine)

In [3]:
```

C. Building a data warehouse for transform superstep.

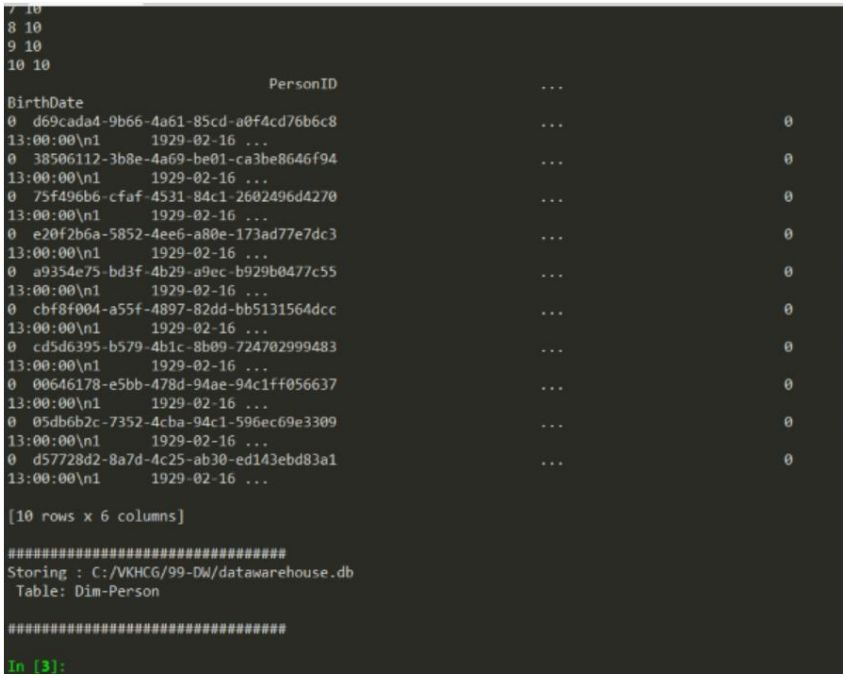
Code:

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



D.Write python program to demonstrate Simple Linear Regression.

Simple Linear Regression

Linear regression is used if there is a relationship or significant association between the variables. This can be checked by scatterplots. If no linear association appears between the variables, fitting a linear regression model to the data will not provide a useful model.

A linear regression line has equations in the following form:

Y = a + bX,

Where, X = explanatory variable and

Y = dependent variable

b = slope of the line

a = intercept (the value of y when x = 0)

Code:

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
import matplotlib.pyplot as plt
import numpy as np
```

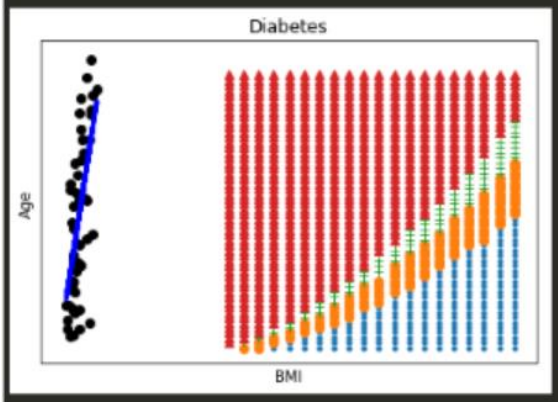
```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
################################################################
Company='01-Vermeulen'
################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
    os.makedirs(sDataVaultDir)
################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
################################################################

t=0
tMax=((300-100)/10)*((300-30)/5)
for heightSelect in range(100,300,10):
    for weightSelect in range(30,300,5):
        height = round(heightSelect/100,3)
        weight = int(weightSelect)
        bmi = weight/(height*height)
        if bmi <= 18.5:
            BMI_Result=1
        elif bmi > 18.5 and bmi < 25:
            BMI_Result=2
        elif bmi > 25 and bmi < 30:
            BMI_Result=3
        elif bmi > 30:
            BMI_Result=4
        else:
```

```
            BMI_Result=0
        PersonLine=[('PersonID', [str(t)]),
                    ('Height', [height]),
                    ('Weight', [weight]),
                    ('bmi', [bmi]),
                    ('Indicator', [BMI_Result])]
        t+=1
        print('Row:',t,'of',tMax)
        if t==1:
            PersonFrame = pd.DataFrame.from_items(PersonLine)
        else:
            PersonRow = pd.DataFrame.from_items(PersonLine)
            PersonFrame = PersonFrame.append(PersonRow)
#################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#################################################################
sTable = 'Transform-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
#################################################################
#################################################################
sTable = 'Person-Satellite-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
#################################################################
sTable = 'Dim-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
#################################################################
fig = plt.figure()
PlotPerson=DimPerson[DimPerson['Indicator']==1]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, ".")
PlotPerson=DimPerson[DimPerson['Indicator']==2]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "o")
PlotPerson=DimPerson[DimPerson['Indicator']==3]
x=PlotPerson['Height']
y=PlotPerson['Weight']
```

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

Output:

```
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

################################
Coefficients:
 [941.43097333]
Mean squared error: 3477.50
Variance score: 0.41
```



Practical 7

AIM: Organizing Data

Theory:

Organize Superstep

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

Horizontal Style

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

A. Write Python program to perform the horizontal style subset of slice of the data warehouse data.

Code:

```
 ################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
################################################################
Company='01-Vermeulen'
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
```

```
#################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#################################################################
print('################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT PersonID,\
       Height,\
       Weight,\
       bmi,\
       Indicator\
  FROM [Dim-BMI]\
  WHERE \
  Height > 1.5 \
  and Indicator = 1\
  ORDER BY  \
       Height,\
       Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#################################################################
sTable = 'Dim-BMI-Horizontal'
print('\n################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('################################')
sTable = 'Dim-BMI-Horizontal'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
```

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

Output:

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

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

In [4]:
```

B. Write Python program to perform the vertical style subset of slice of the data warehouse data.

Vertical Style

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

Code:

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

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

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

Output:

```
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################

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

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

C. Write Python program to perform the island style subset of slice of the data warehouse data.

Island Style

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

Code:

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

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

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

```
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('################################')
#################################################################
```

Output:

```
################################
Working Base : C:/VKHCG  using  win32
################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

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

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

D. Write Python program to perform the secure vault style subset of slice of the data warehouse data and attach the result to the person who performs the query.

Secure Vault Style

The secure vault is a version of one of the horizontal, vertical, or island slicing techniques, but the outcome is also attached to the person who performs the query. This is common in multi-security environments, where different users are allowed to see different data sets.

This process works well, if you use a role-based access control (RBAC) approach to restricting system access to authorized users. The security is applied against the "role," and a person can then, by the security system, simply be added or removed from the role, to enable or disable access.

Code:

```
 ################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
################################################################
Company='01-Vermeulen'
################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
################################################################
print('#################')
```

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

sSQL="SELECT \
       Height,\
       Weight,\
       Indicator,\
       CASE Indicator\
       WHEN 1 THEN 'Pip'\
       WHEN 2 THEN 'Norman'\
       WHEN 3 THEN 'Grant'\
       ELSE 'Sam'\
       END AS Name\
  FROM [Dim-BMI]\
  WHERE Indicator > 2\
  ORDER BY  \
       Height,\
       Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
################################################################
sTable = 'Dim-BMI-Secure'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
print('################################')
sTable = 'Dim-BMI-Secure'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('Only Sam Data')
print(PersonFrame2.head())
print('################################')
################################################################
```

Output:

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

E. Write Python program to demonstrate Association rule mining .

Theory

Association Rule Mining

Association rule learning is a rule-based machine-learning method for discovering interesting relations between variables in large databases, similar to the data you will find in a data lake. The technique enables you to investigate the interaction between data within the same population. Lift is simply estimated by the ratio of the joint probability of two items x and y, divided by the product of their individual probabilities:

Code:

```
import sys

import os

import pandas as pd

from mlxtend.frequent_patterns import apriori

from mlxtend.frequent_patterns import association_rules
```

```
#################################################################
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'
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!! ##############################################')
print('UPGCM')
#####################################################################
```

Output:

```
(541909, 8)
C:\Users\Dell\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:111:
DeprecationWarning: DataFrames with non-bool types result in worse computationalperforma
and their support might be discontinued in the future.Please use a DataFrame with bool t
  warnings.warn(
                  antecedents  ... conviction
0  (ALARM CLOCK BAKELIKE GREEN)  ...   3.791383
1   (ALARM CLOCK BAKELIKE PINK)  ...   3.283859
2  (ALARM CLOCK BAKELIKE GREEN)  ...   4.916181
3    (ALARM CLOCK BAKELIKE RED)  ...   5.568878
4    (ALARM CLOCK BAKELIKE RED)  ...   4.153061

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

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

F. Write Python program to create network routing diagram in organizing superstep .

Theory

Create a Network Routing Diagram

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

Code:

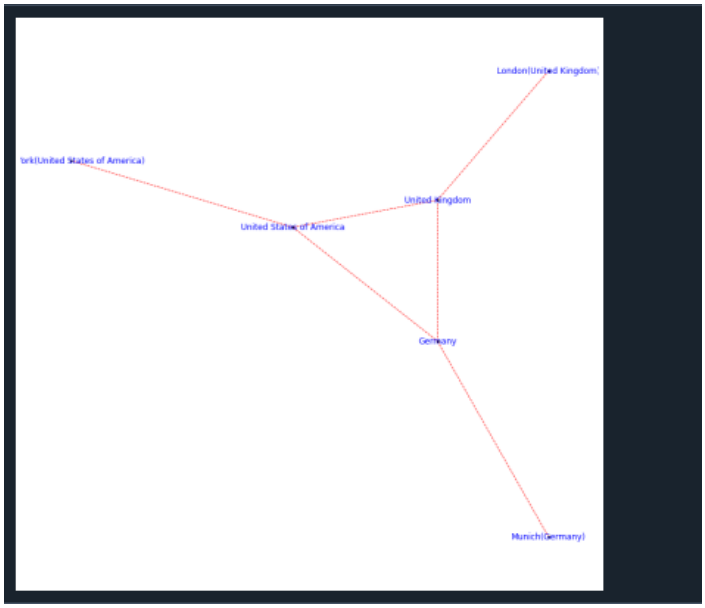
```
#################################################################
import sys

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('UPGCM')

print('Working Base :',Base, ' using ', sys.platform)

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

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

sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-
Company.csv'

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

sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Network-Routing-
Company.gml'

sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Network-Routing-
Company.png'
```

```
Company='01-Vermeulen'
#################################################################
#################################################################
### Import Country Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
print('################################')
#################################################################
print(CompanyData.head())
print(CompanyData.shape)
#################################################################
G=nx.Graph()
for i in range(CompanyData.shape[0]):
    for j in range(CompanyData.shape[0]):
        Node0=CompanyData['Company_Country_Name'][i]
        Node1=CompanyData['Company_Country_Name'][j]
        if Node0 != Node1:
            G.add_edge(Node0,Node1)

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

```
print('Nodes:', G.number_of_nodes())

print('Edges:', G.number_of_edges())

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

sFileName=Base + '/' + Company + '/' + sOutputFileName1

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

print('Storing :',sFileName)

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

nx.write_gml(G, sFileName)

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

sFileName=Base + '/' + Company + '/' + sOutputFileName2

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

print('Storing Graph Image:',sFileName)

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

plt.figure(figsize=(15, 15))

pos=nx.spectral_layout(G,dim=2)

nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)

nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False,
style='dashed')

nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-
serif',font_color='b')

plt.axis('off')

plt.savefig(sFileName,dpi=600)

plt.show()

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

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

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

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


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

Output:

```
In [8]: runfile('C:/Users/Dell/Documents/MSCIT/SEMESTER-I/PRACTICALS/DS/8f.py', wdir='C:/
Users/Dell/Documents/MSCIT/SEMESTER-I/PRACTICALS/DS')
################################
UPGCM
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-
Company.csv
################################
################################
  Company_Country_Code  ...      Company_Country_Name
0                   US  ...  United States of America
1                   US  ...  United States of America
2                   US  ...  United States of America
3                   US  ...  United States of America
4                   US  ...  United States of America

[5 rows x 5 columns]
(150, 5)
Nodes: 6
Edges: 6
################################
Storing : C:/VKHCG/01-Vermeulen/05-Organise/01-EDS/02-Python/Organise-Network-Routing-
Company.gml
################################
```



Practical 8

AIM: Generating Data

Theory:

Report Superstep

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

Vermeulen PLC

Vermeulen requires a map of all their customers' data links

A. Write a python program to perform data visualization to create following graphs using profit data.

Graphics

This section will now guide you through a number of visualizations that particularly useful in presenting data to my customers.

1]Pie Graph

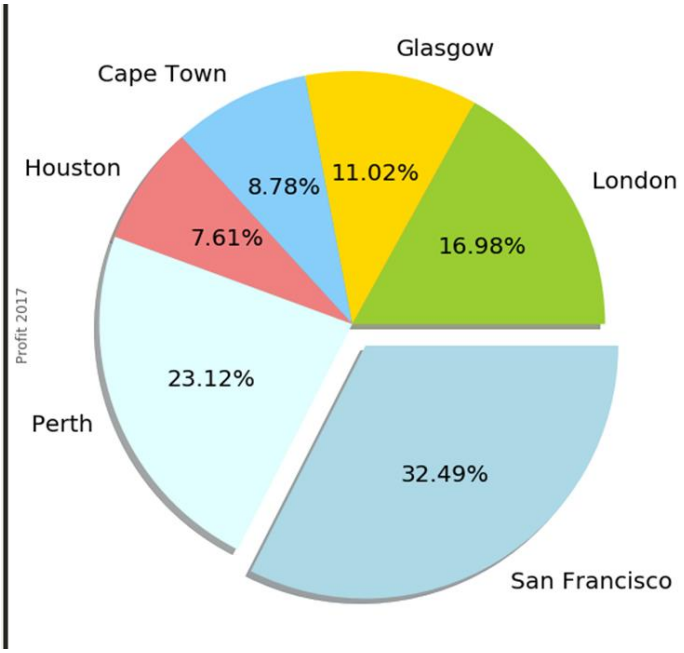
Code:

```
 import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

    Base=os.path.expanduser('~') + '/VKHCG'

else:

      Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)
```

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions',
'ConversionPercent','URL', 'Stats URL'],dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

explode = (0, 0, 0, 0, 0, 0.1)

labels=os_new['Warehouse Location']

colors_mine = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral',

'lightcyan','lightblue']

os_new.plot(figsize=(10, 10),kind="pie", y="Profit 2017",autopct='%.2f%%', \

            shadow=True, explode=explode, legend = False, colors = colors_mine,\

            labels=labels, fontsize=20)

sPicNameOut1=GBase+'pie_explode.png'
```

```
plt.savefig(sPicNameOut1,dpi=600)
```

Output:



2]Double Pie Graph

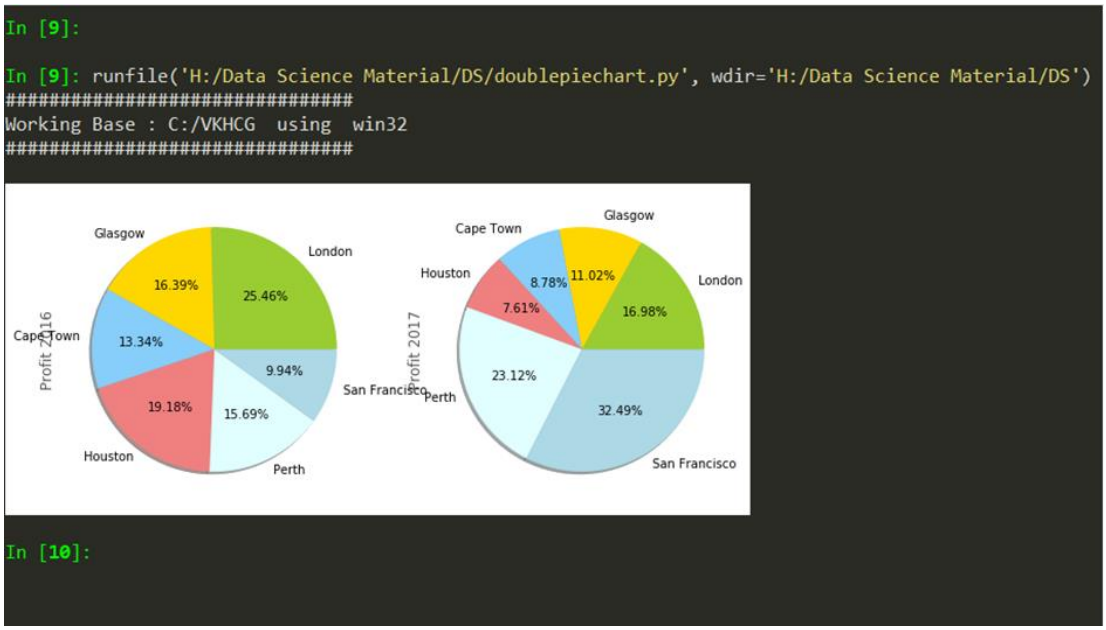
Code:

```
 import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

    Base=os.path.expanduser('~') + '/VKHCG'

else:

      Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)
```

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions',
'ConversionPercent','URL', 'Stats URL'],dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

explode = (0, 0, 0, 0, 0, 0)

colors_mine = ['yellowgreen', 'gold', 'lightskyblue',
'lightcoral','lightcyan','lightblue']

os_new.plot(figsize=(10, 5),kind="pie", y=['Profit 2016','Profit
2017'],autopct='%.2f%%', \

shadow=True, explode=explode, legend = False,colors =

colors_mine,\

subplots=True, labels=labels, fontsize=10)

sPicNameOut2=GBase+'pie.png'
```

```
plt.savefig(sPicNameOut2,dpi=600)
```

Output:



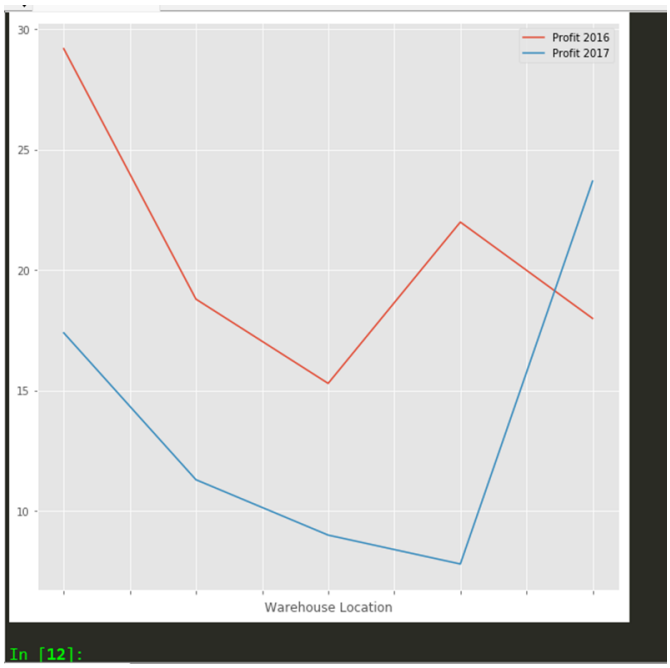
3]Line Graph

Code:

```
 import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

    Base=os.path.expanduser('~') + '/VKHCG'

else:

      Base='C:/VKHCG'

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

```
print('Working Base :',Base, ' using ', sys.platform)

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions',
'ConversionPercent','URL', 'Stats URL'],dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os_new.iloc[:5].plot(figsize=(10, 10),kind='Line',x='Warehouse Location',\

 y=['Profit 2016','Profit 2017']);

sPicNameOut3=GBase+'line.png'

plt.savefig(sPicNameOut3,dpi=600)
```

Output:



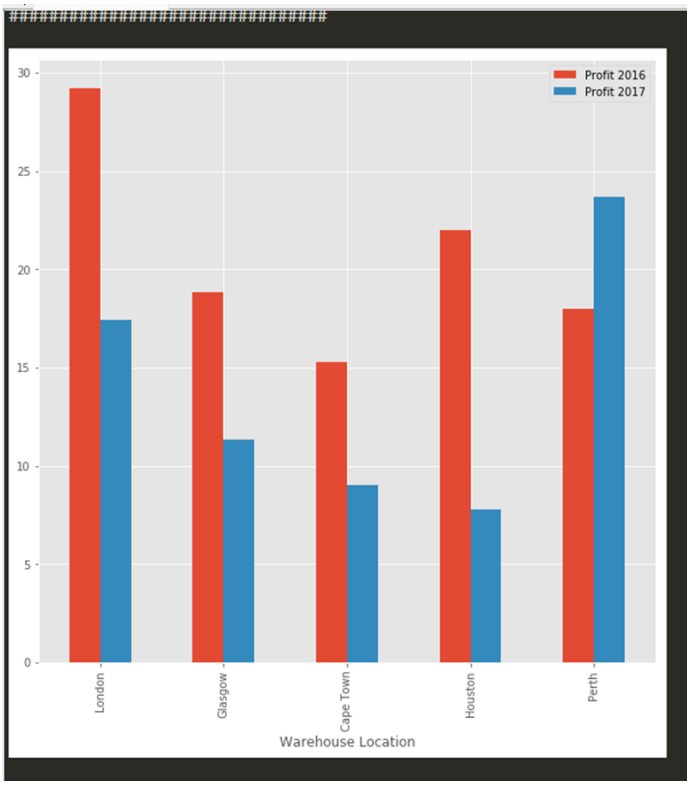
4]Vertical Bar Graph

Code:

```
 import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

    Base=os.path.expanduser('~') + '/VKHCG'

else:

      Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)
```

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions',
'ConversionPercent','URL', 'Stats URL'],dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os_new.iloc[:5].plot(figsize=(10, 10),kind='bar',x='Warehouse Location',\

 y=['Profit 2016','Profit 2017']);

sPicNameOut4=GBase+'bar.png'

plt.savefig(sPicNameOut4,dpi=600)
```

Output:



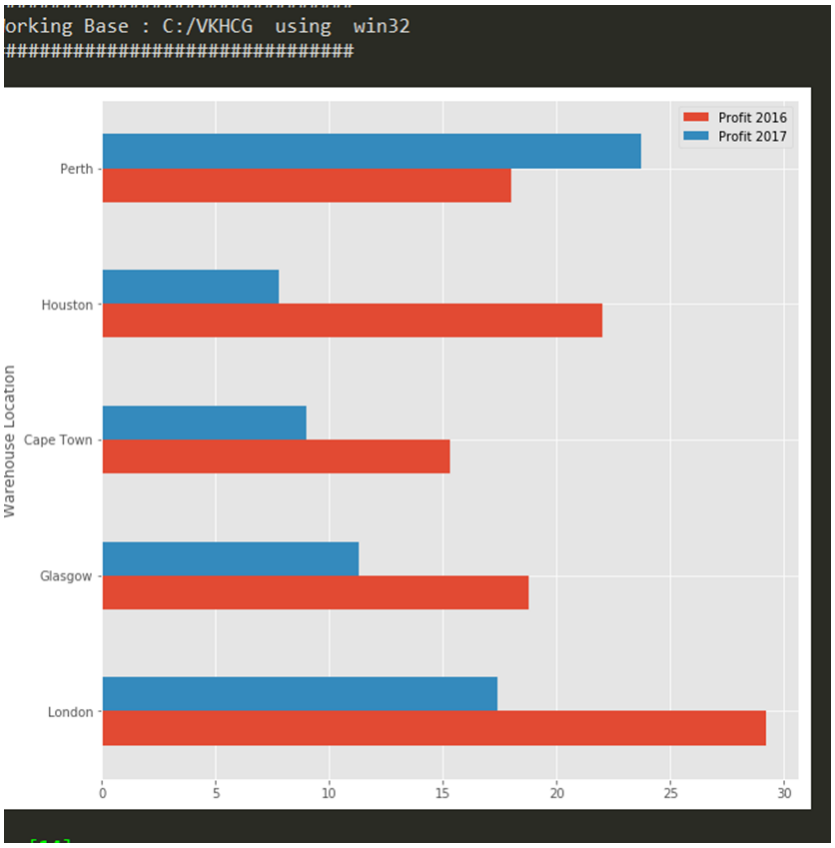
5]Horizontal Bar Graph

Code:

```
 import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

    Base=os.path.expanduser('~') + '/VKHCG'

else:

      Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)
```

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions',
'ConversionPercent','URL', 'Stats URL'],dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os_new.iloc[:5].plot(figsize=(10, 10),kind='barh',x='Warehouse Location',\

 y=['Profit 2016','Profit 2017']);

sPicNameOut5=GBase+'hbar.png'

plt.savefig(sPicNameOut5,dpi=600)
```

Output:



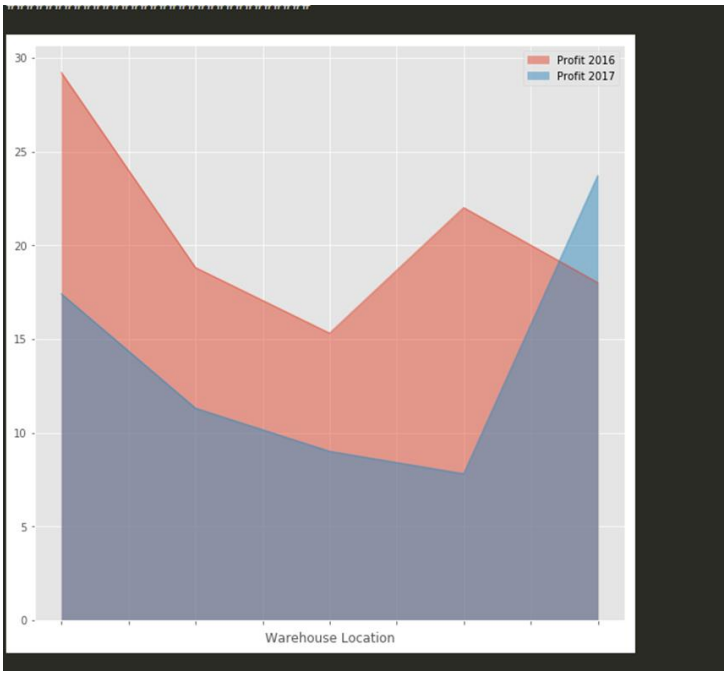
6]Area Graph

Code:

```
 import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

    Base=os.path.expanduser('~') + '/VKHCG'

else:

      Base='C:/VKHCG'
```

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions',
'ConversionPercent','URL', 'Stats URL'],dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os_new.iloc[:5].plot(figsize=(10, 10),kind='area',x='Warehouse Location',\

 y=['Profit 2016','Profit 2017'],stacked=False);

sPicNameOut6=GBase+'area.png'

plt.savefig(sPicNameOut6,dpi=600)
```

Output:



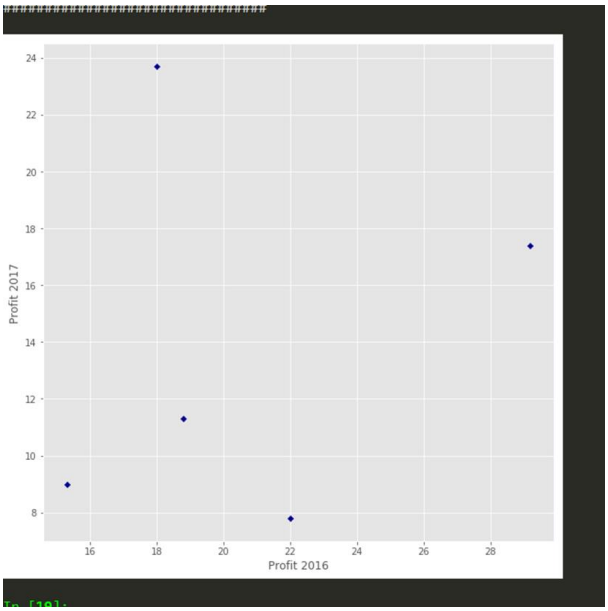
7]Scatter Graph

Code:

```
 import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

    Base=os.path.expanduser('~') + '/VKHCG'

else:

      Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)
```

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions',
'ConversionPercent','URL', 'Stats URL'],dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os_new.iloc[:5].plot(figsize=(10, 10),kind='scatter',x='Profit 2016',\

 y='Profit 2017',color='DarkBlue',marker='D');

sPicNameOut7=GBase+'scatter.png'

plt.savefig(sPicNameOut7,dpi=600)
```

Output:



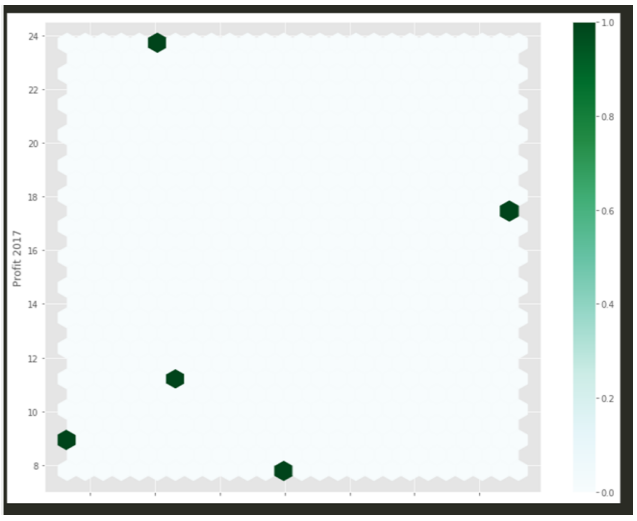
8]Hex Bin Graph

Code:

```
 import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

    Base=os.path.expanduser('~') + '/VKHCG'

else:

      Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]]

os_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions',
'ConversionPercent','URL', 'Stats URL'],dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os_new.iloc[:5].plot(figsize=(13, 10),kind='hexbin',x='Profit 2016',\

 y='Profit 2017', gridsize=25);

sPicNameOut8=GBase+'hexbin.png'

plt.savefig(sPicNameOut8,dpi=600)
```

Output:



B. Write a python program to perform data visualization to create following advanced graphs/plots using profit data.

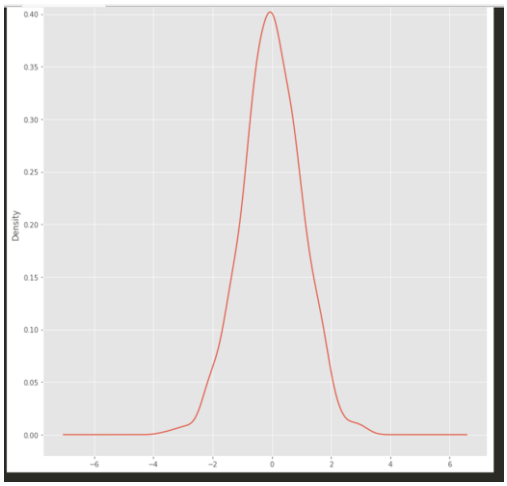
1]Kernel Density Estimation (KDE)Graph

Code:

```
import sys

import os

import pandas as pd

import matplotlib as ml

import numpy as np

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:
```

```
 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

ml.style.use('ggplot')

fig1=plt.figure(figsize=(10, 10))

ser = pd.Series(np.random.randn(1000))

ser.plot(figsize=(10, 10),kind='kde')

sPicNameOut1=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/kde.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight_layout()

plt.show()
```

Output:

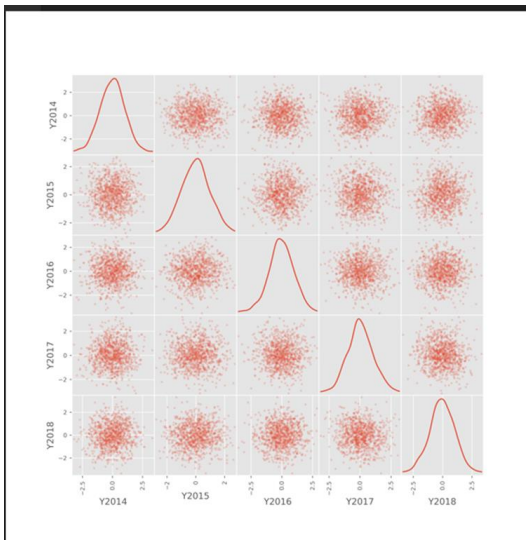


2]Scatter Matrix

Code:

```
import sys

import os

import pandas as pd

import matplotlib as ml

import numpy as np

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:

 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

ml.style.use('ggplot')

fig2=plt.figure(figsize=(10, 10))

from pandas.plotting import scatter_matrix

df = pd.DataFrame(np.random.randn(1000, 5), columns=['Y2014','Y2015',

'Y2016', 'Y2017', 'Y2018'])

scatter_matrix(df, alpha=0.2, figsize=(10, 10), diagonal='kde')

sPicNameOut2=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/scatter_matrix.png'

plt.savefig(sPicNameOut2,dpi=600)

plt.tight_layout()

plt.show()
```

Output:



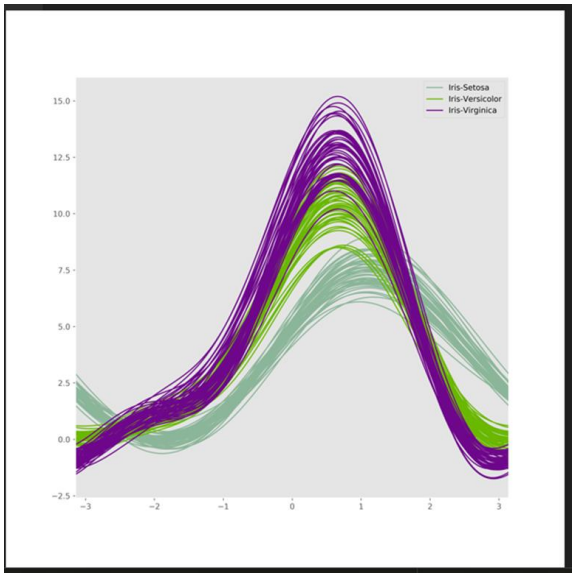
3]Andrew's Curves

Code:

```
import sys

import os

import pandas as pd

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:

 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

sDataFile=Base+'/01-Vermeulen/00-RawData/irisdata.csv'
```

```
data = pd.read_csv(sDataFile)

from pandas.plotting import andrews_curves

plt.figure(figsize=(10, 10))

andrews_curves(data, 'Name')

sPicNameOut1=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/andrews_curves.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight_layout()

plt.show()
```

Output:



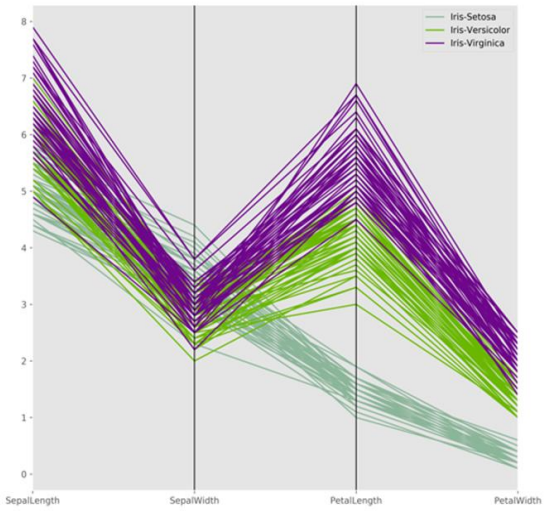
4]Parallel Coordinates

Code:

```
import sys

import os

import pandas as pd

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':
```

```
 Base=os.path.expanduser('~') + '/VKHCG'

else:

 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

sDataFile=Base+'/01-Vermeulen/00-RawData/irisdata.csv'

data = pd.read_csv(sDataFile)

from pandas.plotting import parallel_coordinates

plt.figure(figsize=(10, 10))

parallel_coordinates(data, 'Name')

sPicNameOut2=Base+'/01-Vermeulen/06-Report/01-EDS/02-
Python/parallel_coordinates.png'

plt.savefig(sPicNameOut2,dpi=600)

plt.tight_layout()

plt.show()
```

Output:

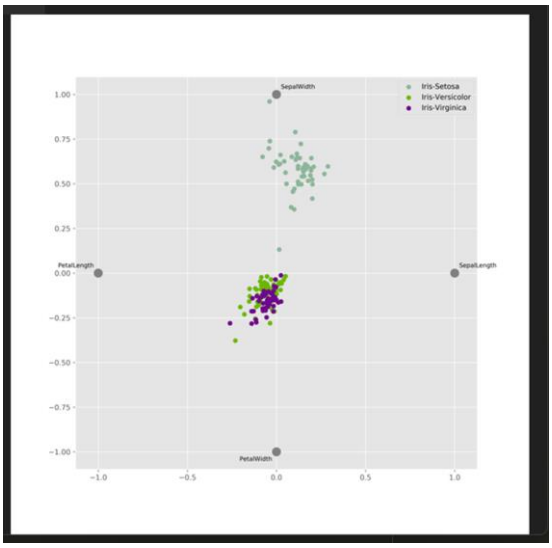


5]RADVIZ method

Code:

```
import sys

import os

import pandas as pd

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:

 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

sDataFile=Base+'/01-Vermeulen/00-RawData/irisdata.csv'

data = pd.read_csv(sDataFile)

from pandas.plotting import radviz

plt.figure(figsize=(10, 10))

radviz(data, 'Name')

sPicNameOut3=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/radviz.png'

plt.savefig(sPicNameOut3,dpi=600)

plt.tight_layout()

plt.show()
```

Output:



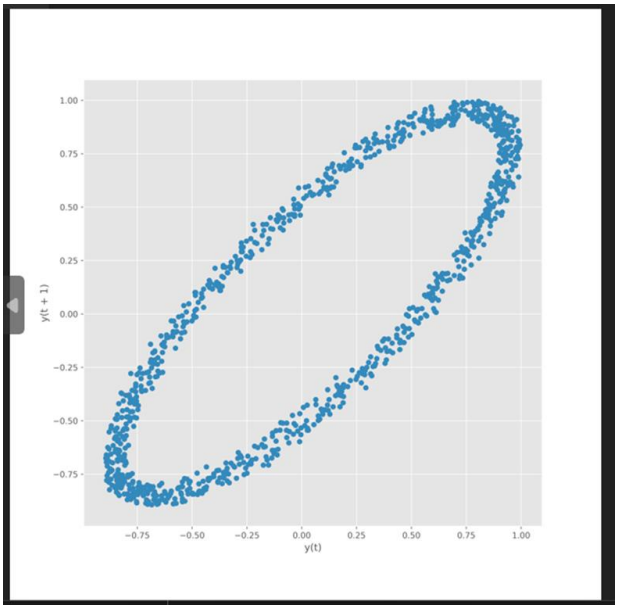
6]Lag plot

Code:

```
import sys

import os

import pandas as pd

from matplotlib import style

from matplotlib import pyplot as plt

import numpy as np

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

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:

 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

style.use('ggplot')

from pandas.plotting import lag_plot

plt.figure(figsize=(10, 10))

data = pd.Series(0.1 * np.random.rand(1000) + \

 0.9 * np.sin(np.linspace(-99 * np.pi, 99 * np.pi, num=1000)))

lag_plot(data)

sPicNameOut1=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/lag_plot.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight_layout()

plt.show()
```

Output:



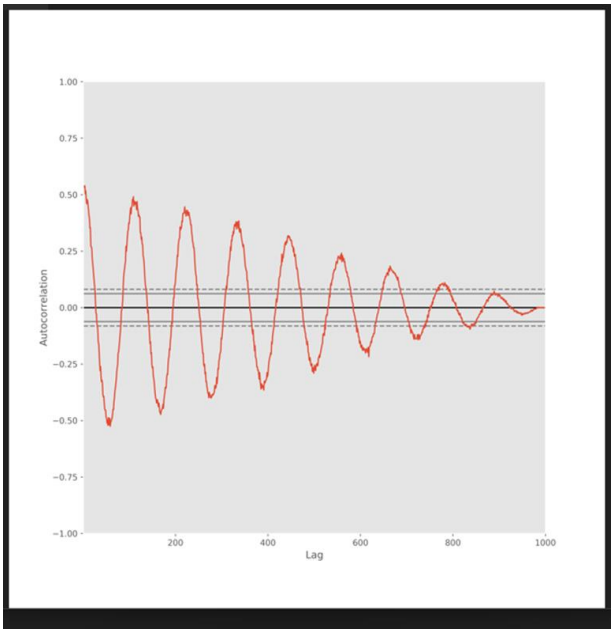
7]Autocorrelation Plot

Code:

```
 import sys

import os

import pandas as pd

from matplotlib import style

from matplotlib import pyplot as plt

import numpy as np

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

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:

 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

style.use('ggplot')

from pandas.plotting import autocorrelation_plot

plt.figure(figsize=(10, 10))

data = pd.Series(0.7 * np.random.rand(1000) + \

 0.3 * np.sin(np.linspace(-9 * np.pi, 9 * np.pi, num=1000)))

autocorrelation_plot(data)

sPicNameOut2=Base+'/01-Vermeulen/06-Report/01-EDS/02-
Python/autocorrelation_plot.png'

plt.savefig(sPicNameOut2,dpi=600)

plt.tight_layout()
```

```
plt.show()
```

Output:



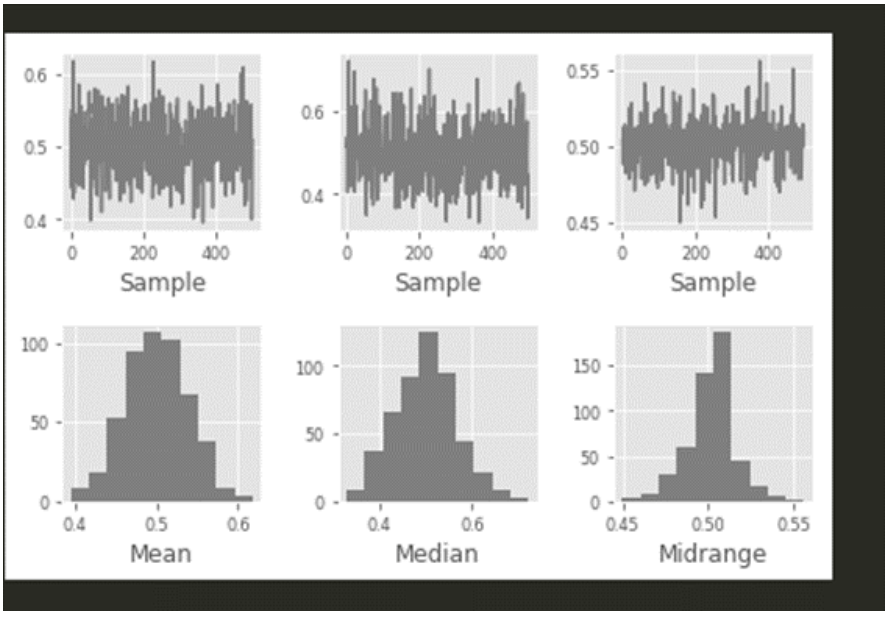
8]Bootstrap Plot

Code:

```
import sys

import os

import pandas as pd

from matplotlib import style

from matplotlib import pyplot as plt

import numpy as np

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

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:
```

```
 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

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

style.use('ggplot')

from pandas.plotting import bootstrap_plot

data = pd.Series(np.random.rand(1000))

plt.figure(figsize=(10, 10))

bootstrap_plot(data, size=50, samples=500, color='grey')

sPicNameOut3=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/bootstrap_plot.png'

plt.savefig(sPicNameOut3,dpi=600)

plt.tight_layout()

plt.show()
```

Output:



9]Contour Graphs

Code:

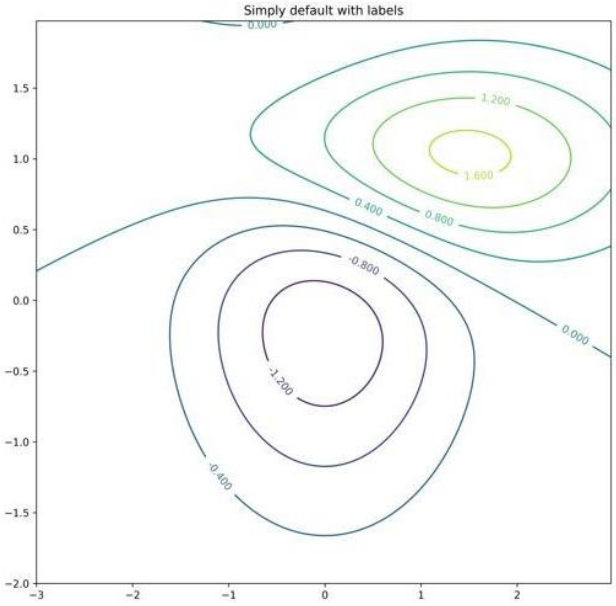
```
 import sys

import os

import matplotlib

import numpy as np

import matplotlib.cm as cm

import matplotlib.mlab as mlab

import matplotlib.pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:

 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

```
matplotlib.rcParams['xtick.direction'] = 'out'

matplotlib.rcParams['ytick.direction'] = 'out'

delta = 0.025

x = np.arange(-3.0, 3.0, delta)

y = np.arange(-2.0, 2.0, delta)

X, Y = np.meshgrid(x, y)

Z1 = mlab.bivariate_normal(X, Y, 1.0, 1.0, 0.0, 0.0)

Z2 = mlab.bivariate_normal(X, Y, 1.5, 0.5, 1, 1)
```

```
# difference of Gaussians

Z = 10.0 * (Z2 - Z1)

plt.figure(figsize=(10, 10))

CS = plt.contour(X, Y, Z)

plt.clabel(CS, inline=1, fontsize=10)

plt.title('Simply default with labels')

sPicNameOut0=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/contour0.png'

plt.savefig(sPicNameOut0,dpi=600)

plt.tight_layout()

plt.show()

10. 3D Graph

import sys

import os

import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

from sklearn import decomposition

from sklearn import datasets

if sys.platform == 'linux' or sys.platform == ' darwin':

 Base=os.path.expanduser('~') + '/VKHCG'

else:

 Base='C:/VKHCG'

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

print('Working Base :',Base, ' using ', sys.platform)

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

np.random.seed(5)
```

```
centers = [[1, 1], [-1, -1], [1, -1]]

iris = datasets.load_iris()

X = iris.data

y = iris.target

fig = plt.figure(1, figsize=(16, 12))

plt.clf()

ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

plt.cla()

pca = decomposition.PCA(n_components=3)

pca.fit(X)

X = pca.transform(X)

for name, label in [('Setosa', 0), ('Versicolour', 1), ('Virginica', 2)]:

      ax.text3D(X[y == label, 0].mean(),

           X[y == label, 1].mean() + 1.5,

            X[y == label, 2].mean(), name,

              horizontalalignment='center',

 bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))

y = np.choose(y, [1, 2, 0]).astype(np.float)

ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y,
cmap=plt.cm.Spectral,edgecolor='k',marker='p',s=300)

ax.w_xaxis.set_ticklabels([])

ax.w_yaxis.set_ticklabels([])

ax.w_zaxis.set_ticklabels([])

sPicNameOut0=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/3DPlot.png'

plt.savefig(sPicNameOut0,dpi=600)

plt.show()
```

Output:



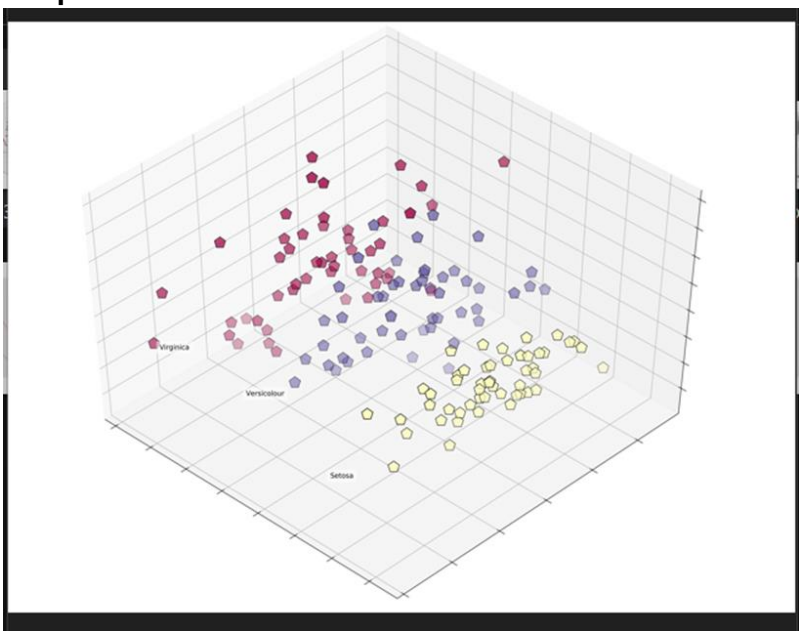
10]3D Graphs

Code:

```
######################################################################
# -*- coding: utf-8 -*-
######################################################################
import sys
import os
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import decomposition
from sklearn import datasets
######################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
######################################################################
np.random.seed(5)

centers = [[1, 1], [-1, -1], [1, -1]]
iris = datasets.load_iris()
X = iris.data
```

```
y = iris.target

fig = plt.figure(1, figsize=(16, 12))
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

plt.cla()
pca = decomposition.PCA(n_components=3)
pca.fit(X)
X = pca.transform(X)

for name, label in [('Setosa', 0), ('Versicolour', 1), ('Virginica', 2)]:
    ax.text3D(X[y == label, 0].mean(),
              X[y == label, 1].mean() + 1.5,
              X[y == label, 2].mean(), name,
              horizontalalignment='center',
              bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))
# Reorder the labels to have colors matching the cluster results
y = np.choose(y, [1, 2, 0]).astype(np.float)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y, cmap=plt.cm.spectral,
           edgecolor='k',marker='p',s=300)

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])


sPicNameOut0=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/3DPlot.png'
plt.savefig(sPicNameOut0,dpi=600)
plt.show()
```

Output:



Practical 9

AIM: Data Visualization with Power BI

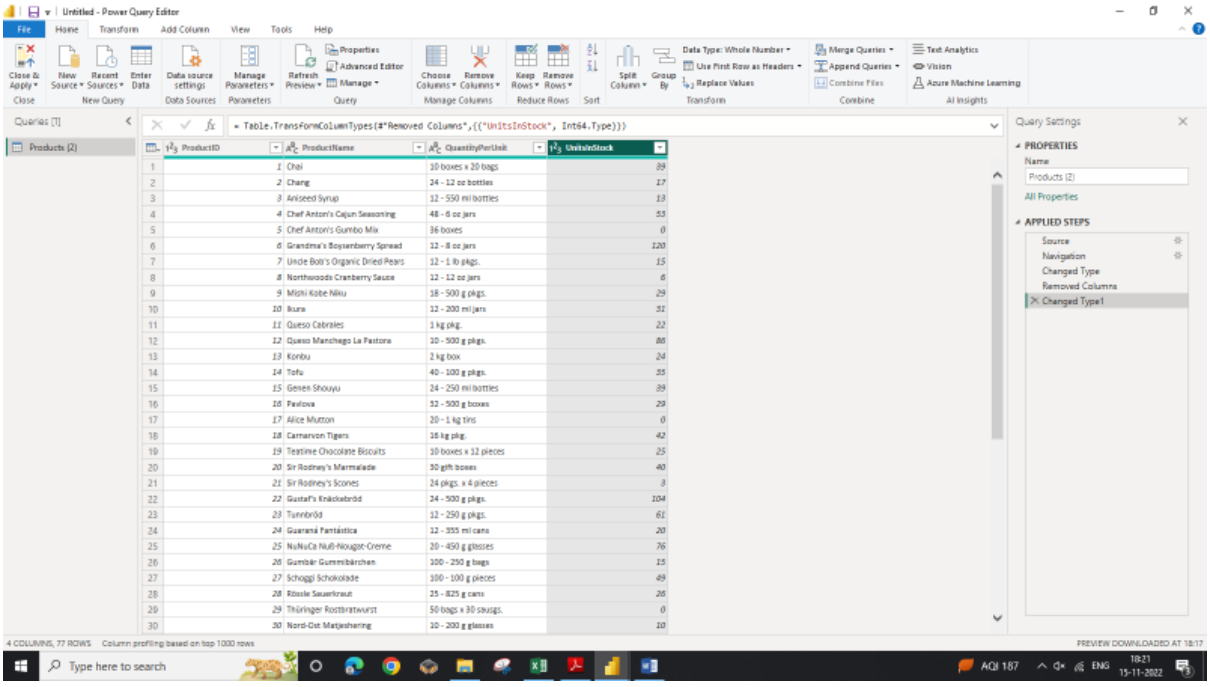
Case Study: Sales Data

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 Collections, 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.

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

1. In Query Editor, select the ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns *(use Ctrl+ Click to select more than one columns , 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 columns 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

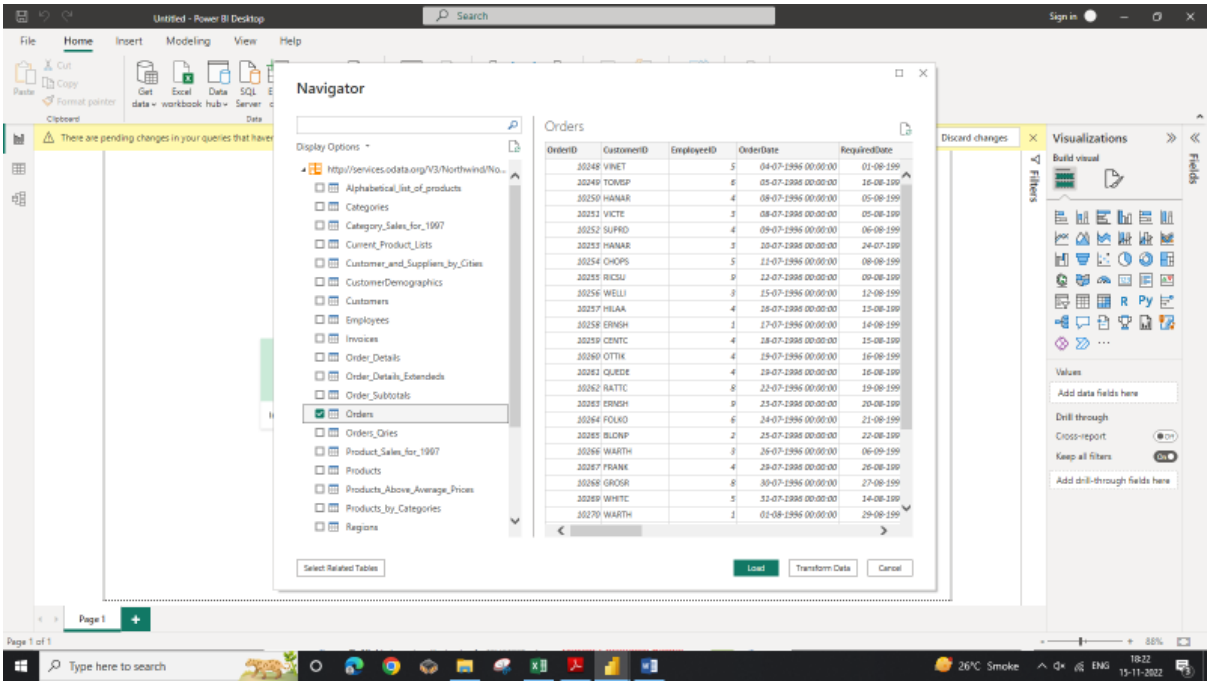
3. If not already a Whole Number, set Whole Number for data type from the drop down *(the Data Type: button also displays the data type for the current selection).*

Task 2: Import order data from an OData feed

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

Step 1: Connect to an OData feed

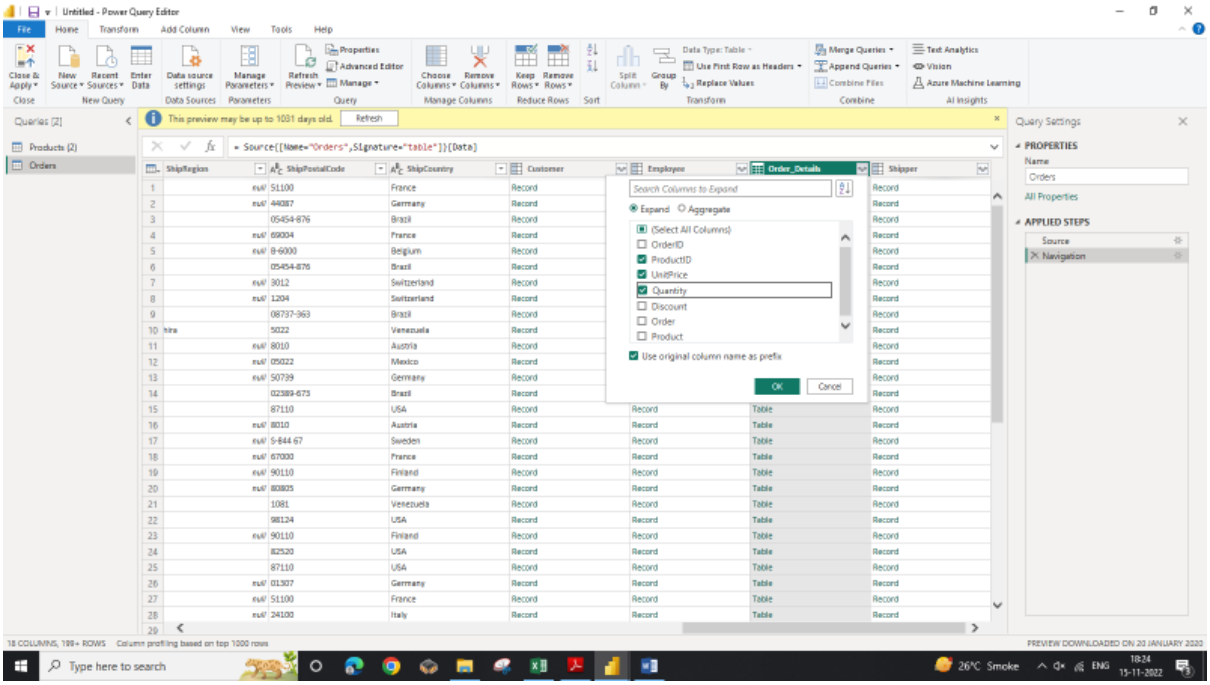
1. From the **Home** ribbon tab in Query Editor, select **Get Data**.
2. Browse to the **OData Feed** data source.
3. In the **OData Feed** dialog box, paste the **URL** for the Northwind OData feed.
4. Select **OK**.



Step 2: Expand the Order_Details table

Expand the **Order_Details** table that is related to the **Orders** table, to combine the **ProductID, UnitPrice**, and **Quantity** columns from **Order_Details** into the **Orders** table.
The **Expand** operation combines columns from a related table into a subject table. When the query runs, rows from the related table **(Order_Details)** are combined into rows from the subject table **(Orders).**

After you expand the **Order_Details** table, three new columns and additional rows are added to the **Orders** table, one for each row in the nested or related table.
1. In the **Query View,** scroll to the **Order_Details** column.
2. In the **Order_Details** column, select the expand icon ().
3. In the **Expand** drop-down: a. Select **(Select All Columns)** to clear all columns.
Select **ProductID, UnitPrice,** and **Quantity.**
click **OK**.



Step 3: Remove other columns to only display columns of interest
In this step you remove all columns except **OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice,** and **Order_Details.Quantity** columns. In the previous task, you used **Remove Other Columns**. For this task, you remove selected columns.
In the **Query View**, select all columns by completing a.
a. Click the first column **(OrderID)**.
b. Shift+Click the last column **(Shipper**).
c. Now that all columns are selected, use Ctrl+Click to unselect the following columns:
OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and

Order_Details.Quantity.

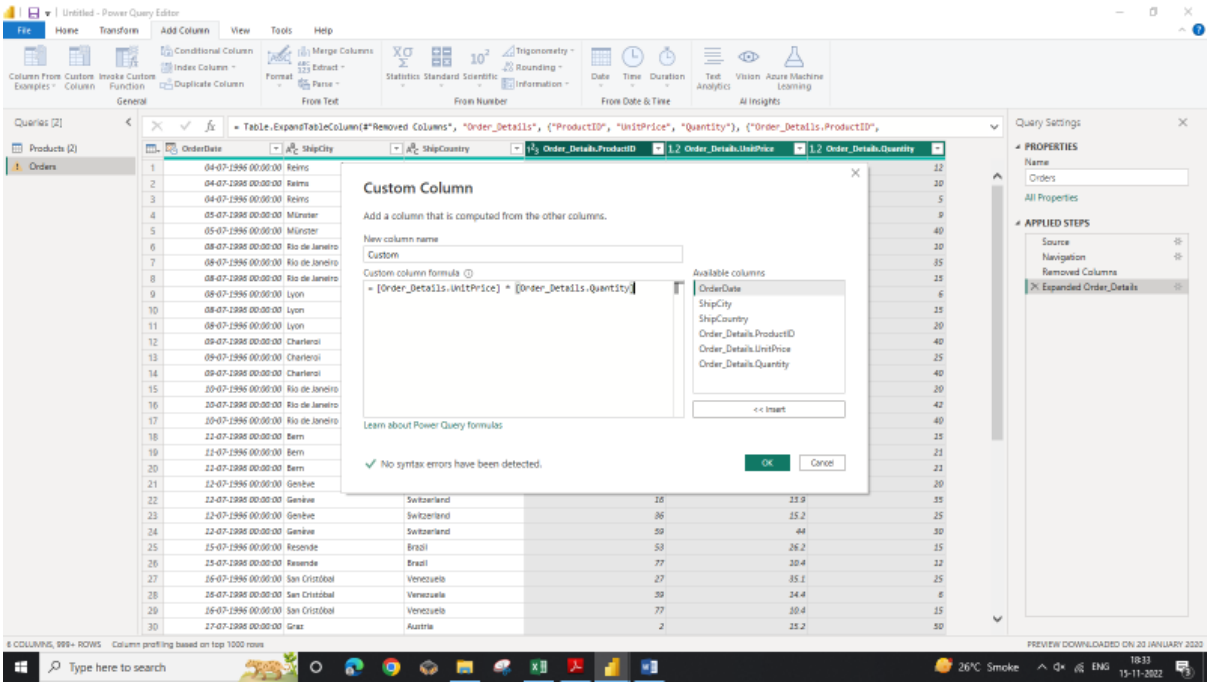
Now that only the columns we want to remove are selected, right-click on any selected column header and click Remove Columns.

Step 4: Calculate the line total for each Order_Details row

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

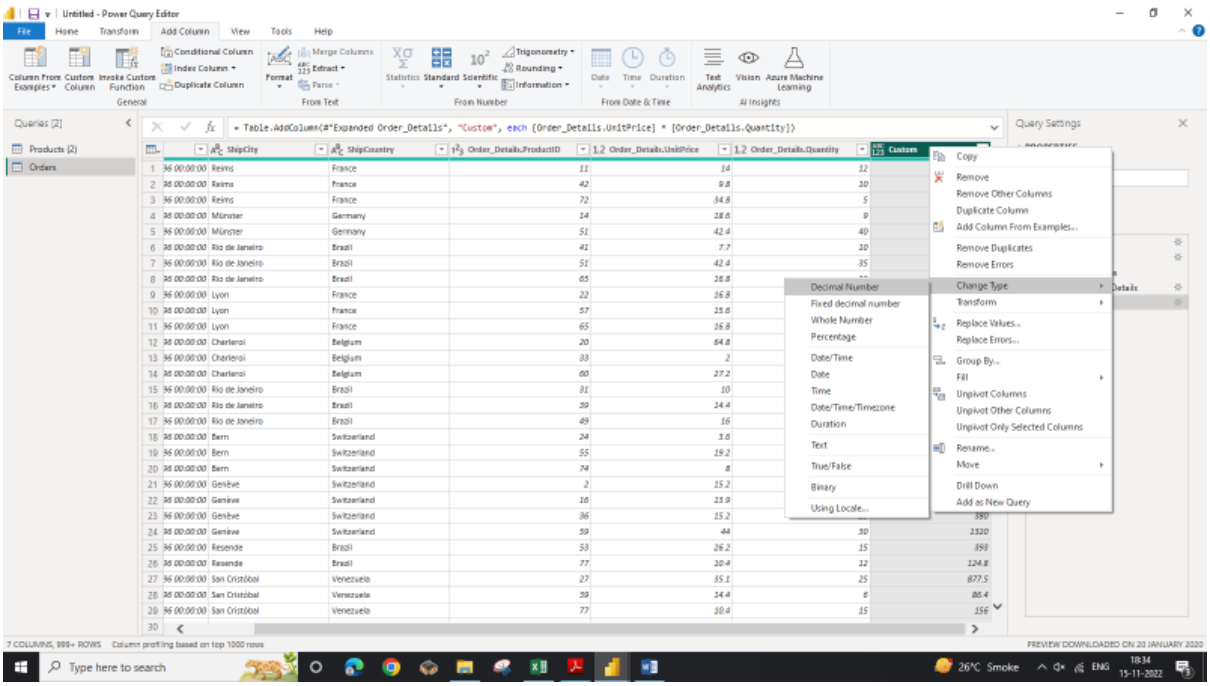
Calculate the line total for each **Order_Details** row:

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



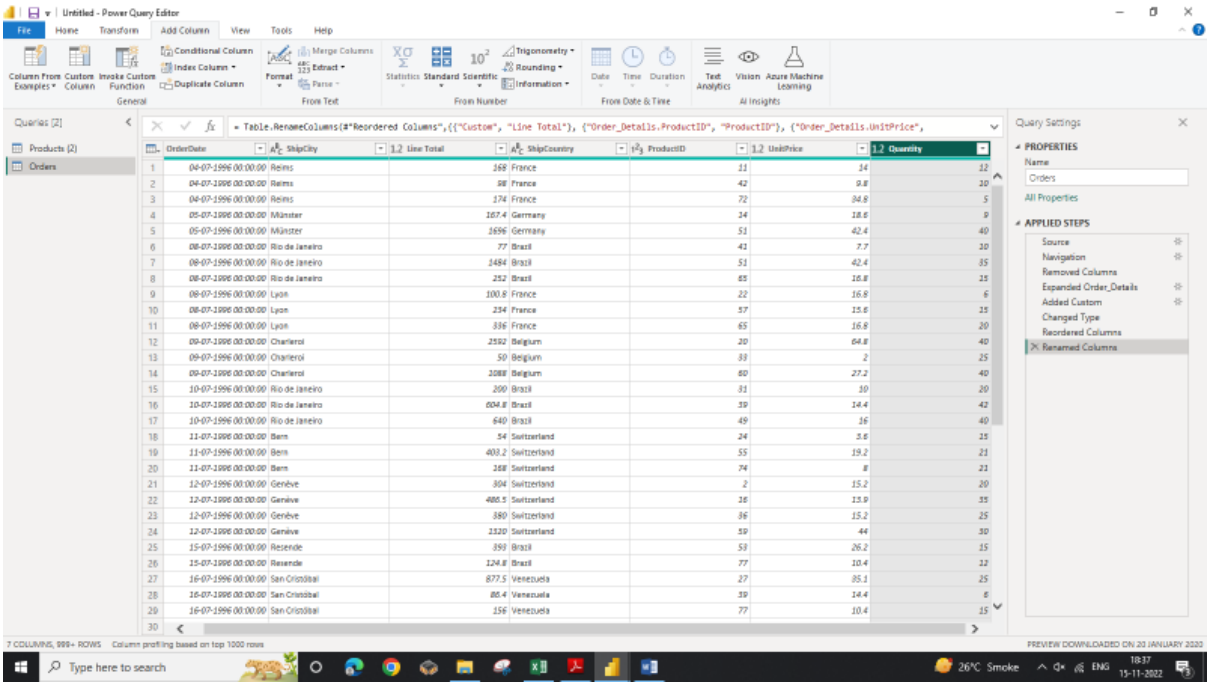
Step 5: Set the datatype of the LineTotal field

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



Step 6: Rename and reorder columns in the query

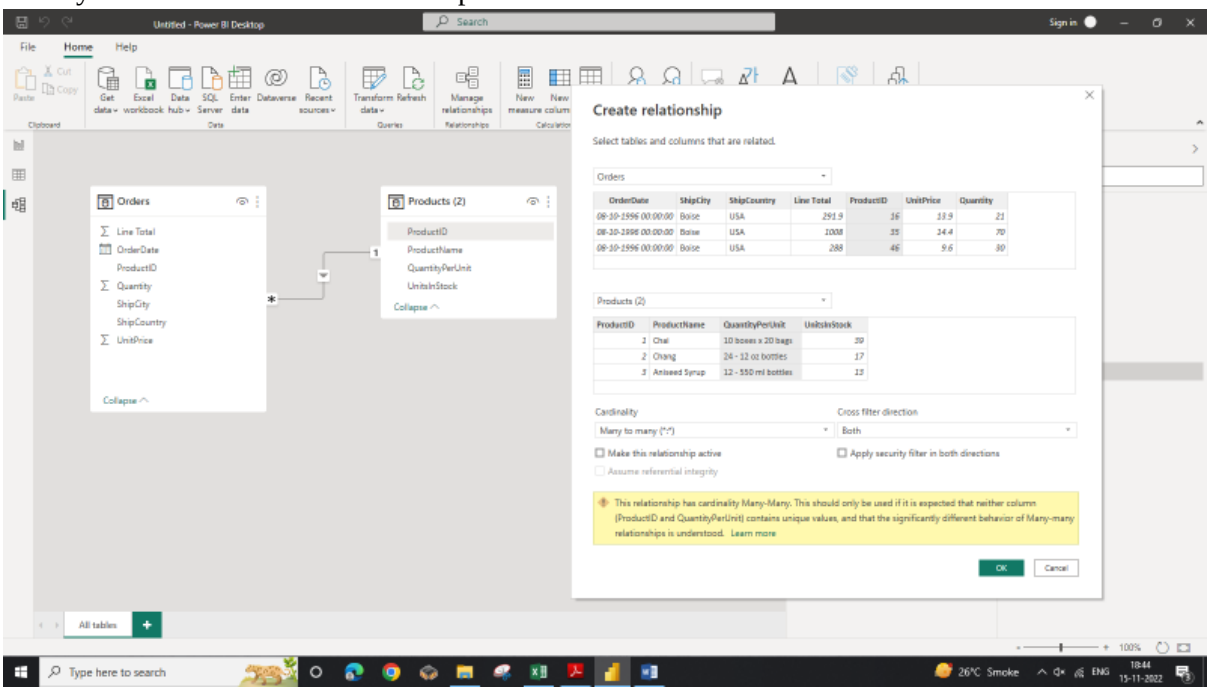
1. In **Query Editor,** drag the **LineTotal** column to the left, after **ShipCountry.**
2. Remove



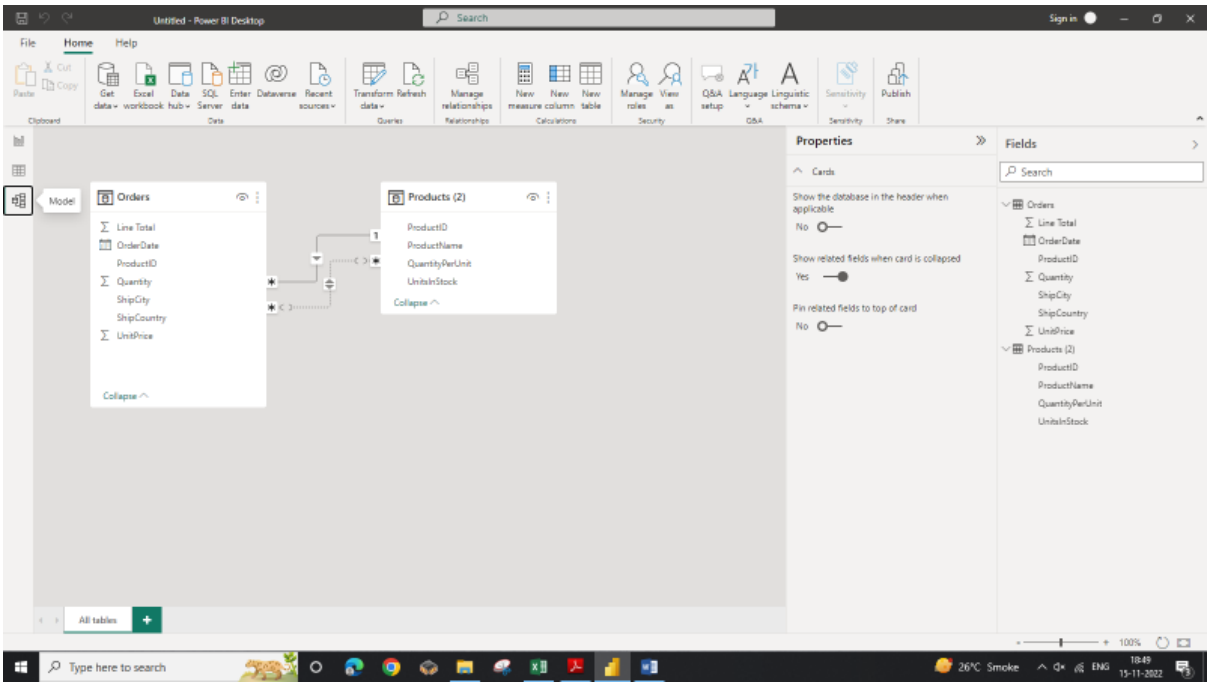
2. Remove the Order_Details. prefix from the **Order_Details.ProductID, Order_Details.UnitPrice** and **Order_Details.Quantity** columns, by double-clicking on each column header, and then deleting that text from the column name.

Task 3: Combine the Products and Total Sales queries

1.First, we need to load the model that we created in Query Editor into Power BI Deshtop. From the **Home** Ribbon of Query Editor, select **Close & Load**.
2. Power BI Desktop loads the data from the two queries
3. Once the data is loaded, select the Manage Relationships button Home ribbon
4. Select the New… button
5. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.

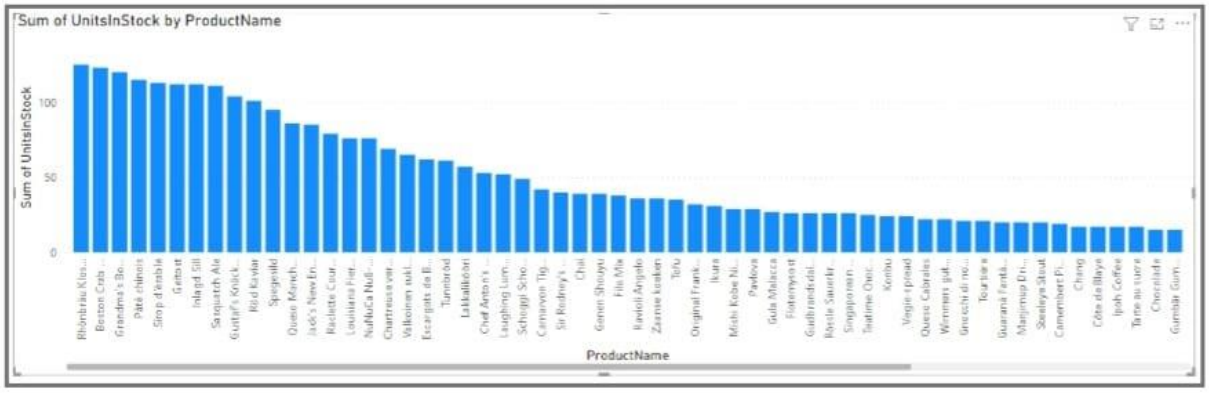


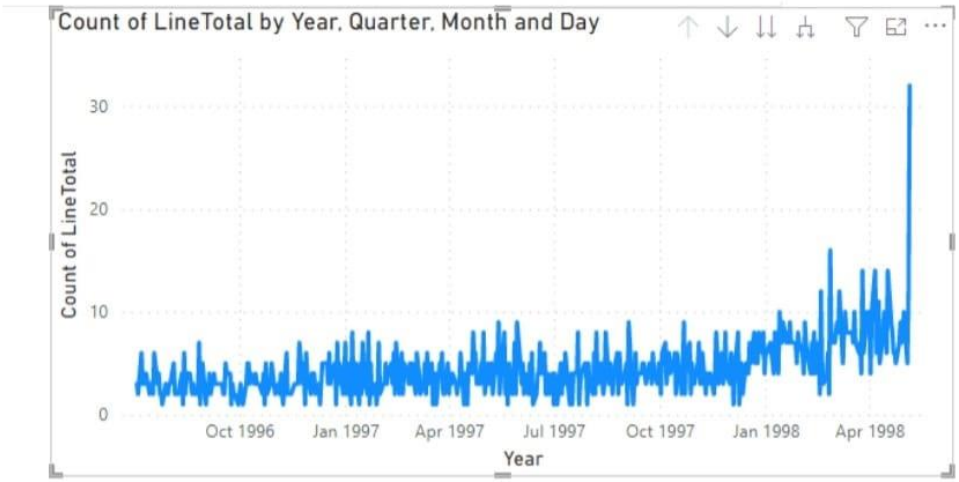
6.Select Cancel, and then select Relationship view in Power BI Desktop.



Task 4: Build visuals using your data

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





Step 2: Drag OrderDate to the canvas beneath the first chart, then drag LineTotal (again, the Fields pane) onto the visual, then select Line Chart. The following visualization is created.

Step 3: Next, drag ShipCountry to a spare 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.