VSIT

DATA SCIENCE
Practical # 1

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	FY M.Sc. IT
Topic	Assessing Data	**Division**	A

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

i. Drop the Columns Where All Elements Are Missing Values

Common Steps:

Step 1: Import packages (sys, os, pandas)
Step 2: Create a base directory with the file location, preferably 'C:/'
Step 3: Create your file and folder paths for given files respectively (company, VKHCG path, etc.) and create a variable merging the entire path with Base directory.
Step 4: Check whether the above-mentioned directory is created or not, if not use a function 'os.mkdirs' to create the directory.
Step 5: Read the csv file and print raw values

Step for 'Data where column is missing all values':

Step 6: Use function '.dropna' and mention axis = 1 as well as how = 'all' to drop all the columns who have only missing/null values

```
import sys
import os
import pandas as pd
##############Enter required path############################
Base='C:/VKHCG'
#################################################################
print("Ninad Karlekar 22306A1012")
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
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("Ninad Karlekar 22306A1012")
print('### Done!! ####################')
print('#################################')
```

```
Ninad Karlekar 22306A1012
################################
Working Base : C:/VKHCG  using  win32
################################
################################
## Raw Data Values
################################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
################################
################################
## Data Profile
################################
Rows : 21
Columns : 8
################################
################################
```

```
################################
## Test Data Values
################################
      ID FieldA  FieldB FieldC  FieldD    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0   10241.0       1
1    2.0   Good     NaN   Best   512.0    5121.0       2
2    3.0   Good  Better    NaN   256.0     256.0       3
3    4.0   Good  Better   Best     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0    6411.0       5
5    6.0   Good     NaN   Best    32.0      32.0       6
6    7.0    NaN  Better   Best    16.0    1611.0       7
7    8.0    NaN     NaN   Best     8.0    8111.0       8
8    9.0    NaN     NaN    NaN     4.0      41.0       9
9   10.0      A       B      C     2.0   21111.0      10
10   NaN    NaN     NaN    NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0  102411.0      12
12  10.0   Good     NaN   Best   512.0     512.0      13
13  10.0   Good  Better    NaN   256.0    1256.0      14
14  10.0   Good  Better   Best     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     164.0      16
16  10.0   Good     NaN   Best    32.0     322.0      17
17  10.0    NaN  Better   Best    16.0     163.0      18
18  10.0    NaN     NaN   Best     8.0     844.0      19
19  10.0    NaN     NaN    NaN     4.0    4555.0      20
20  10.0      A       B      C     2.0     111.0      21
################################
## Data Profile
################################
Rows : 21
Columns : 7
################################
################################
Ninad Karlekar 22306A1012
### Done!! ####################
################################
```

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

Step for 'Data where column is missing any elements':

Use function '.dropna' and mention axis = 1 as well as how = 'any' to drop all the columns who have only missing/null values

```
print("Ninad Karlekar 22306A1012")
import sys
import os
import pandas as pd
##############Enter required path############################
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
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("Ninad Karlekar 22306A1012")
print('### Done!! ####################')
```

```
In [12]: runfile('F:/MSC IT/Practical/DS/
Ninad Karlekar 22306A1012
#################################
Working Base : C:/VKHCG  using  win32
#################################
## Raw Data Values
##############################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
##############################
## Data Profile
#############################
Rows : 21
Columns : 8
```

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

```
## Data Profile
################################
Rows : 21
Columns : 1
################################
################################
Ninad Karlekar 22306A1012
### Done!! ####################
################################

In [13]:
```

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

Steps for 'Keeping rows with maximum two values missing':

Use function '.dropna' and mention thresh = (no. of column - 2) to keep all the rows who have maximum of 2 missing/null values

```
print("Ninad Karlekar 22306A1012")
import sys
import os
import pandas as pd
##############Enter required path############################
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
RawData=pd.read_csv(sFileName,header=0)
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=6)
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("Ninad Karlekar 22306A1012")
print('### Done!! ####################')
print('#################################')
#####################################################################
```

```
In [14]: runfile('F:/MSc IT/Practical/DS/Code Files/Prac_1__A_III
Ninad Karlekar 22306A1012
#################################
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
```

```
## 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
```

```
## 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
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
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
20  10.0      A       B      C     2.0     NaN     111.0      21
################################
## Data Profile
```

```
20  10.0      A       B       C
################################
## Data Profile
################################
Rows : 15
Columns : 8
################################
Ninad Karlekar 22306A1012
### Done!! ####################
################################
```

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

```
# Ninad Karlekar 22306A1012
# Write Python / R program to create the network routing diagram from the given data on routers.
########## Assess-Network-Routing-Company.py ######################
```

```
import sys
import os
import pandas as pd
pd.options.mode.chained_assignment = None
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using Windows')
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(sFileName)

print('Loading :',sFileName)
print('##################################')
CountryData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)

## 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)

### 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)

## 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)
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('Changed :',CustomerData.columns.values)
CustomerData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CustomerData.columns.values)
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!! ######################')
```

```
Ninad Karlekar 22306A1012
################################
Working Base : C:/VKHCG  using Windows
C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv
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']
################################
 'Country_Name']
################################
To  ['Company_Country_Code' 'Company_Place_Name' 'Company_Latitude'
 'Longitude' 'Country_Name']
################################
To  ['Company_Country_Code' 'Company_Place_Name' 'Company_Latitude'
 'Company_Longitude' 'Country_Name']
################################
To  ['Company_Country_Code' 'Company_Place_Name' 'Company_Latitude'
 'Company_Longitude' 'Company_Country_Name']
################################
################################
Storing : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv
################################
Ninad Karlekar 22306A1012
### Done!! ####################
```

VSIT

DATA SCIENCE
Practical # 2

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Conversion different data format to HORUS format	**Division**	A

A. Text delimited CSV to HORUS format.

Step 1:- Import Libraries
Step 2:- Give the path of the csv file
Step 3:- Remove the Column 'ISO-2-Code' and 'ISO-3-Code'
Step 4:- Rename 'Country' as 'CountryName' and 'ISO-M49' as 'CountryNumber'
Step 5:- Set 'CountryNumber' as new Index
Step 6:- Sorting the 'CountryName'
Step 7:- Print and Save the Output Data to csv

```
# Utility Start CSV to HORUS ================================
# Standard Tools
print("Ninad Karlekar 22306A1012")
print("Text delimited CSV to HORUS format.")
import pandas as pd
# Input Agreement ============================================
sInputFileName=r"F:\MSC IT\Practical\DS\Prac2Ninad\Country_Code.csv"
InputData = pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
# 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('CountryName',inplace=True)
# Sort data by CurrencyName
ProcessData.sort_values('CountryName',axis=0,ascending=False,inplace=True)
print("**********************************************************************************")
print("ProcessData")
print(ProcessData)
OutputData=ProcessData
sOutputFileName =r"F:\MSC IT\Practical\DS\Prac2Ninad\HORUSCountry_Code.csv"
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
```

```
In [17]: runfile('F:/MSC IT/Practical/DS/Prac2Ninad/prac3A.py',
Ninad Karlekar 22306A1012
Text delimited CSV to HORUS format.
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]
************************************************************
```

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

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

B. JSON to HORUS Format

Step 1:- Import Libraries
Step 2:- Give the path of the json file
Step 3:- Remove the Column 'ISO-2-Code' and 'ISO-3-Code'
Step 4:- Rename 'Country' as 'CountryName' and 'ISO-M49' as 'CountryNumber'
Step 5:- Set 'CountryNumber' as new Index
Step 6:- Sorting the 'CountryName'
Step 7:- Print and Save the Output Data to csv

```
# Utility Start JSON to HORUS ================================
# Standard Tools
#=============================================================
print("Ninad Karlekar 22306A1012")
print("C.        JSON to HORUS Format")
import pandas as pd
# Input Agreement ============================================
sInputFileName=r"F:\MSC IT\Practical\DS\Prac2Ninad\Country_Code.json"
InputData=pd.read_json(sInputFileName, orient='index', encoding="latin-1")
print('Input Data Values ==================================')
print(InputData)
print('=====================================================')
# Processing Rules ===========================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CountryName
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values ================================')
```

```
print(ProcessData)
print('================================================')
# Output Agreement ============================================
OutputData=ProcessData
sOutputFileName=r"F:\MSC IT\Practical\DS\Prac2Ninad\HORUS-JSON-Country.csv"
OutputData.to_csv(sOutputFileName, index = False)
print('JSON to HORUS - Done')
# Utility done ================================================
```

```
In [30]: runfile('F:/MSC IT/Practical/DS/Prac2Ninad/prac3C.py',
Prac2Ninad')
Ninad Karlekar 22306A1012
C.  JSON to HORUS Format
Input Data Values ==================================
                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0                 Afghanistan         AF        AFG        4
1               Aland Islands         AX        ALA      248
2                     Albania         AL        ALB        8
3                     Algeria         DZ        DZA       12
4              American Samoa         AS        ASM       16
..                        ...        ...        ...      ...
242  Wallis and Futuna Islands         WF        WLF      876
243            Western Sahara         EH        ESH      732
244                     Yemen         YE        YEM      887
245                    Zambia         ZM        ZMB      894
246                  Zimbabwe         ZW        ZWE      716

[247 rows x 4 columns]
================================================
```

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

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

In [31]:
```

VSIT

DATA SCIENCE
Practical # 3

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Conversion different data format to HORUS format	**Division**	A

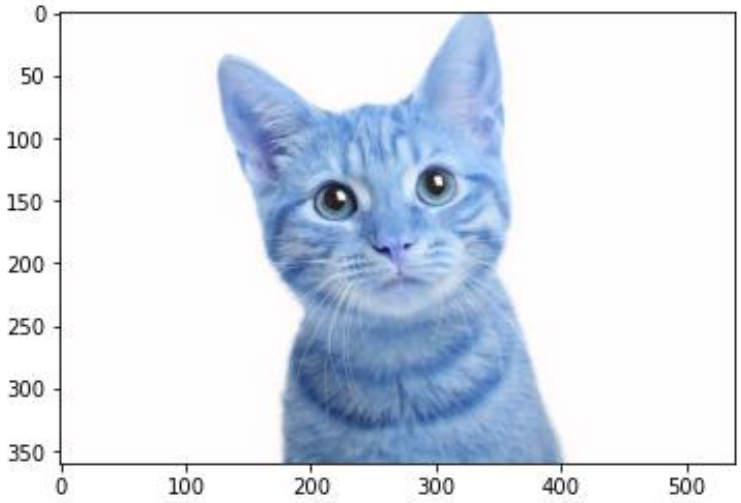
C. IMG to HORUS Format

1. We need to install packages such as image and with_statement in order to import the package.
2. In the next step, we load the image into the program.
3. Our system stores the pixels information so that we can access them in the future.
4. We also assign tuples x and y dimensions.
5. It is necessary to create a CSV file in order to store the pixel's information, such as its RGB colours.
6. At last, hexadecimal values of colour are appended to this csv file.

```
import cv2 as cv
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Input Agreement ========================================
sInputFileName='F:/MSC IT/Practical/DS/prac3/content/d.jpg'
InputData = cv.imread(sInputFileName,cv.IMREAD_COLOR)
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']
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='F:/MSC IT/Practical/DS/prac3/content/d1.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=================================================')
print('Picture to HORUS - Done')
print('=================================================')
```

```
In [26]: runfile('F:/MSC IT/Pr
prac3')
Input Data Values ============
X:  360
Y:  540
RGBA:  3
==============================
Rows:  116640
Columns : 5
==============================
Process Data Values ==========
==============================
==============================
Storing File
==============================
Picture to HORUS - Done
==============================

In [27]:
```



D. Conversion from Video/Audio to HORUS

1. To import packages, we need packages such as os, shutil.
2. After loading the image, we run the program
3. We store the information for further access.
4. Further, we assign x and y dimensions of the image to tuple.
5. A CSV file is created to store the information.

```
from __future__ import with_statement
from PIL import Image # pip install Pillow
import cv2 # pip install opencv-python
print("Ninad Karlekar 22306A1012")
vidcap = cv2.VideoCapture('C:/VKHCG/05-DS/9999-Data/dog.mp4')
success,image = vidcap.read()
count = 0
while success:
  cv2.imwrite("C:/VKHCG/05-DS/9999-Data/temp/frame%d.jpg" % count, image)     # save frame as
JPEG file
  success,image = vidcap.read()
  print('Read a new frame: ', success)
  count += 1
# Part 2: Frames to Horus

num = 0
with open('Video-to-HORUS-output_fileF.csv', 'a+') as f:
    f.write('R,G,B,FrameNumber\n')
for c in range(count):
    #print('C:/VKHCG/05-DS/9999-Data/temp/frame%d.jpg'%num)
    im = Image.open('C:/VKHCG/05-DS/9999-Data/temp/frame%d.jpg'%num)
```

```
        pix = im.load()
        width, height = im.size
        with open('Video-to-HORUS-output_fileF.csv', 'a+') as f:
            for x in range(width-300):
                for y in range(height-300):
                    r = pix[x,y][0]
                    g = pix[x,x][1]
                    b = pix[x,x][2]
                    f.write('{0},{1},{2},{3}\n'.format(r,g,b,num))

        num = num + 1

    print('Movie to Frames HORUS - Done')
    print('====================================================')
    # Utility done ==============================================
    print("Ninad Karlekar 22306A1012")
```

```
In [28]: runfile('F:/MSC IT/Pract
Ninad Karlekar 22306A1012
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
Read a new frame:  True
```

E. XML to HORUS Format

1. To import libraries.
2. Define a function which will take data frame and convert that into xml data
3. Define a function which will convert xml to df
4. Create a variable and use an xml file in it.
5. Process data after processing data drop column ISO-2-code and ISO-3-code.
6. inplace=True will reflect the changes in the csv.
7. Set-index is used to create a new column.
8. Then sort data by currency no.

```
# Utility Start XML to HORUS ================================
# Standard Tools
print("Ninad Karlekar 22306A1012")
print("B.        XML to HORUS Format")
```

```
import pandas as pd
import xml.etree.ElementTree as ET
def df2xml(data):
    header = data.columns
    root = ET.Element('root')
    for row in range(data.shape[0]):
        entry = ET.SubElement(root, 'entry')
        for index in range(data.shape[1]):
            schild = str(header[index])
            child = ET.SubElement(entry, schild)
            if str(data[schild][row]) != 'nan':
                child.text = str(data[schild][row])
            else:
                child.text = 'n/a'
            entry.append(child)
    result = ET.tostring(root)
    return result

def xml2df(xml_data):
    root = ET.XML(xml_data)
    all_records = []
    for i, child in enumerate(root):
        record = {}
        for subchild in child:
            record[subchild.tag] = subchild.text
        all_records.append(record)
    return pd.DataFrame(all_records)
sInputFileName = r"F:\MSC IT\Practical\DS\Prac2Ninad\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 CurrencyNumberProcessData.sort_values('CountryName', axis=0, ascending=False,
inplace=True)
print('=====================================================')
print('Process Data Values ================================')
print('=====================================================')
print(ProcessData)
print('=====================================================')
OutputData = ProcessData
sOutputFileName = r"F:\MSC IT\Practical\DS\Prac2Ninad\HORUS-XML-Country.csv"
OutputData.to_csv(sOutputFileName, index=False)
print('=====================================================')
print('XML to HORUS - Done')
print('=====================================================')
# Utility done ================================================
```

```
In [44]: runfile('F:/MSC IT/Practical/DS/Prac2Ninad/prac3B.py', wdir='F:/MSC IT/Practical/DS/
Prac2Ninad')
Ninad Karlekar 22306A1012
B.  XML to HORUS Format
=====================================================
Input Data Values ==================================
=====================================================
<root><entry><Country>Afghanistan</Country><Country>Afghanistan</Country><ISO-2-CODE>AF</ISO-2-
CODE><ISO-2-CODE>AF</ISO-2-CODE><ISO-3-Code>AFG</ISO-3-Code><ISO-3-Code>AFG</ISO-3-Code><ISO-
M49>4</ISO-M49><ISO-M49>4</ISO-M49></entry><entry><Country>Aland Islands</Country><Country>Aland
Islands</Country><ISO-2-CODE>AX</ISO-2-CODE><ISO-2-CODE>AX</ISO-2-CODE><ISO-3-Code>ALA</ISO-3-
Code><ISO-3-Code>ALA</ISO-3-Code><ISO-M49>248</ISO-M49><ISO-M49>248</ISO-M49></
entry><entry><Country>Albania</Country><Country>Albania</Country><ISO-2-CODE>AL</ISO-2-
```

```
Country><Country>Zimbabwe</Country><ISO-2-CODE>ZW</ISO-2-CODE><ISO-2-CODE>ZW</ISO-2-CODE><ISO-3-
Code>ZWE</ISO-3-Code><ISO-3-Code>ZWE</ISO-3-Code><ISO-M49>716</ISO-M49><ISO-M49>716</ISO-M49></
entry></root>
=====================================================
=====================================================
Process Data Values ================================
=====================================================
                          CountryName
CountryNumber
4                          Afghanistan
248                       Aland Islands
8                              Albania
12                             Algeria
16                      American Samoa
...                                ...
876           Wallis and Futuna Islands
732                     Western Sahara
887                              Yemen
894                             Zambia
716                           Zimbabwe

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

In [45]:
```

VSIT

DATA SCIENCE
Practical # 4

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Utilities and Auditing	**Batch**	1

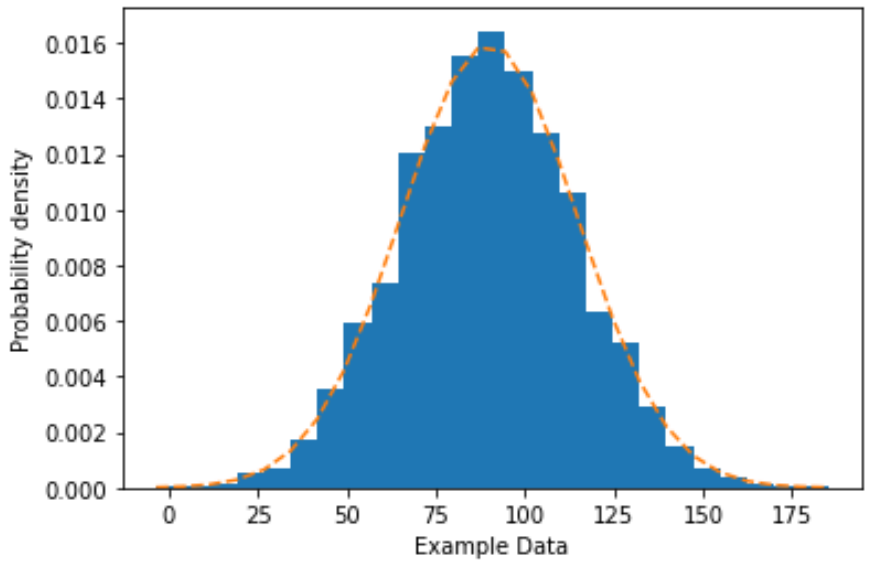
A. Fixers Utilities:

```
#--------------------------- Program to Demonstrate Fixers utilities ------------------
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print('#1 Removing leading or lagging spaces from a data entry');
baddata = " Hello My name is Ninad Karlekar "
print('>',baddata,'<')
cleandata=baddata.strip()
print('>',cleandata,'<')
print("**********************************************************")
# 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)
print("**********************************************************")
# 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(2002, 1, 9)
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)
```

```
In [34]: runfile('F:/MSC IT/Practical/DS/prac3/new/Utitlites_1_1.py', wdir='F:/MSC IT/
Practical/DS/prac3/new')
#1 Removing leading or lagging spaces from a data entry
>  Hello My name is Ninad Karlekar  <
> Hello My name is Ninad Karlekar <
**********************************************************
#2 Removing nonprintable characters from a data entry
Bad Data :  Data Science with funny characters is bad!!!
Clean Data :  DataScience with funny characters is bad!!!
**********************************************************
# 3 Reformatting data entry to match specific formatting criteria.
Bad Data :  2002-01-09
Good Data :  09 January 2002

In [35]:
```

B. Data Binning or Bucketing

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



C. Averaging of Data

```
import pandas as pd
#######################################################################
InputFileName='IP_DATA_CORE.csv'
```

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

```
In [43]: runfile('F:/MSC IT/Practical/DS/prac3/new/Utitlites_3
Practical/DS/prac3/new')
################################
Working Base : C:/VKHCG  using
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
     Country Place_Name  Latitude
0         US   New York   40.7528
1         US   New York   40.7528
2         US   New York   40.7528
3         US   New York   40.7528
4         US   New York   40.7528
...      ...        ...       ...
3557      DE     Munich   48.0915
3558      DE     Munich   48.1833
3559      DE     Munich   48.1000
3560      DE     Munich   48.1480
3561      DE     Munich   48.1480

[3562 rows x 3 columns]
Country  Place_Name
DE       Munich        48.143223
GB       London        51.509406
US       New York      40.747044
Name: Latitude, dtype: float64

In [44]:
```

D. Outlier Detection

```
import pandas as pd
################################################################
```

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

```
In [45]: runfile('F:/MSC IT/Practica
Practical/DS/prac3/new')
################################
Working Base : C:/VKHCG
################################
Loading : C:/VKHCG/01-Vermeulen/00-R
All Data
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
...      ...        ...       ...
3434      GB     London   51.5092
3435      GB     London   51.5092
3436      GB     London   51.5163
3437      GB     London   51.5085
3438      GB     London   51.5136

[1502 rows x 3 columns]
```

```
Lower than  51.5061708702166
     Country Place_Name  Latitude
1915      GB     London   51.4739
Not Outliers
     Country Place_Name  Latitude
1917      GB     London   51.5085
1918      GB     London   51.5085
1922      GB     London   51.5085
1928      GB     London   51.5085
1929      GB     London   51.5085
...      ...        ...       ...
3432      GB     London   51.5092
3433      GB     London   51.5092
3434      GB     London   51.5092
3435      GB     London   51.5092
3437      GB     London   51.5085

[1485 rows x 3 columns]

In [46]:
```

E. Logging

```
import sys    import os
import logging
import uuid
import shutil
import time
print("Ninad Karlekar 22306A1012")
Base='C:/Spyder Practials'
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()
        for hdlr in log.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)
        logging.basicConfig(level=logging.DEBUG,format='%(asctime)s %(name)-12s %(levelname)-8s
%(message)s',datefmt='%m-%d %H:%M', filename=sLogFile, filemode='w')
        console = logging.StreamHandler()
        console.setLevel(logging.INFO)
        formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
        console.setFormatter(formatter)
        logging.getLogger('').addHandler(console)
        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.')
```

```
print("Ninad Karlekar 22306A1012")
```

```
In [20]: runfile('F:/MSc IT/Practical/DS/Code Files/Prac_4__2.py', wdir='F:/MSc IT/Practical/DS/Code Files')
Ninad Karlekar 22306A1012
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:/Spyder Practials/01-Vermeulen/01-Retrieve/Logging/Logging_557fdaf5-0bd4-4e05-9048-1e2da91b3f26.log
root        : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-02-Assess-info: INFO     Practical Data Science logged information message.
Apllication-01-Vermeulen-02-Assess-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01-Vermeulen-02-Assess-error: ERROR    Practical Data Science logged an error message.
Set up: C:/Spyder Practials/01-Vermeulen/02-Assess/Logging/Logging_03072048-3af1-4b05-9514-8b312b09674a.log
root        : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-03-Process-info: INFO     Practical Data Science logged information message.
Apllication-01-Vermeulen-03-Process-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01-Vermeulen-03-Process-error: ERROR    Practical Data Science logged an error message.
Set up: C:/Spyder Practials/01-Vermeulen/03-Process/Logging/Logging_43fb51c4-e343-4f3d-8d5f-3866a8f222c3.log
root        : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-04-Transform-info: INFO     Practical Data Science logged information message.
Apllication-01-Vermeulen-04-Transform-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-04-Transform-error: ERROR    Practical Data Science logged an error message.
Set up: C:/Spyder Practials/04-Clark/04-Transform/Logging/Logging_84aeb16e-6780-416b-a06d-6958ce92b9f1.log
root        : INFO     Practical Data Science is fun!.
Apllication-04-Clark-05-Organise-info: INFO     Practical Data Science logged information message.
Apllication-04-Clark-05-Organise-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-05-Organise-error: ERROR    Practical Data Science logged an error message.
Set up: C:/Spyder Practials/04-Clark/05-Organise/Logging/Logging_c3d151ab-91ee-44b9-baa3-017c281db2ae.log
root        : INFO     Practical Data Science is fun!.
Apllication-04-Clark-06-Report-info: INFO     Practical Data Science logged information message.
Apllication-04-Clark-06-Report-warning: WARNING  Practical Data Science logged a warning message.
Apllication-04-Clark-06-Report-error: ERROR    Practical Data Science logged an error message.
Set up: C:/Spyder Practials/04-Clark/06-Report/Logging/Logging_8de697df-4305-42d1-865f-8ef8c6232e3e.log
Ninad Karlekar 22306A1012

In [21]:
```

VSIT

DATA SCIENCE
Practical # 5

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Retrieving Data	**Batch**	1

Topic: Program to retrieve different types of attributes

Step 1: Import libraries. [sys, os, pandas]
Step 2: Define the base path and name of the input file.
Step 3: Read the CSV file into a dataframe using pandas.
Step 4: Print the raw input data.
Step 5: Process the data by replacing the whitespaces with a period “.”
Step 6: Print the modified columns.

```
import sys
import os
import pandas as pd
print("Ninad Karlekar 22306A1012")
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("Ninad Karlekar 22306A1012")
print('### Done!! ############################################')
#####################################################################
```

```
Ninad Karlekar 22306A1012
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'>
```

```
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
Ninad Karlekar 22306A1012
### Done!! ########################
```

Data Pattern

Pre-requisite library installation:

1. readr:
   install.packages('readr', dependencies = TRUE, repos='http://cran.rstudio.com/')
   2. data.table
   install.packages("data.table", dependencies=TRUE)

Step 1: Import libraries. [readr, data.table]
Step 2: Define the file name and location.
Step 3: Read the CSV file into a dataframe using pandas.
Step 4: Create a data table object
Step 5: Create a new dataframe to store modified table
Step 6: Replace letters with “A”, digits with “N” and white spaces with “.”
Step 7: Display the modified table

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

```
  };
  for (n in seq(0,9,1)){
    s=chartr(as.character(n),"N",s)
  };
  s=chartr(" ","b",s)
  s=chartr(".","u",s)
  pattern_country[r,]$PatternCountry=s;
};
View(pattern_country)
```

	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
11	*NA*	*NA*
12	NG	AA
13	SD	AA
14	ZM	AA
15	TZ	AA
16	ZA	AA

Loading IP_DATA_ALL

Step 1: Import libraries. [sys, os, pandas]
Step 2: Define the base path and name of the input file.
Step 3: Read the CSV file into a dataframe using pandas.
Step 4: Print the raw input data.
Step 5: Process the data by replacing the whitespaces with a period “.”
Step 6: Print the modified columns.
Step 7: Print the FIXED data rows.

```
import sys
import os
import pandas as pd
print("Ninad Karlekar 22306A1012")
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("Ninad Karlekar 22306A1012")
#####################################################################
print('### Done!! ############################################')
```

```
In [9]: runfile('F:/MSC IT/Practical/DS/Code files/Prac_5__C.
DS/Code files')
Ninad Karlekar 22306A1012
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
Ninad Karlekar 22306A1012
### Done!! ############################################
```

VSIT

DATA SCIENCE
Practical # 6

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Organizing Data	**Batch**	1

Horizontal style

Step 1: Import sys, os, pandas as pd, and sqlite3 as sq libraries
Step 2: Create a variable named 'Base' and assign it the value 'C:/VKHCG'
Step 3: Print the working base and platform you are using with the help of the sys.platform
Step 4: Create a variable named 'Company' and assign it the value '01-Vermeulen'
Step 5: Create a variable to store the warehouse directory and set the value as Base + '/99-DW'. this is the location where you will store your warehouse data
Step 6: Check if the DataWarehouse directory exists or not if it does not exist then create it using the method 'makedirs' like os.makedirs(sDataWarehouse)
Step 7: Create a variable to store a database name and assign it to a value as sDataWarehouseDir + '/datawarehouse.db'
Step 8: Now create a connection variable 'conn1' to store the connection with the datawarehouse database using the sq.connect(databasename_variable) method
Step 9: Create another variable to store the datamart database and assign it a value as sDataWarehouseDir + '/datamart.db'
Step 10: Create connection variable conn2 to store connection with the datamart database using the sq.connect() method
Step 11: Create a variable to table name from which we are going to fetch data and assign it 'Dim-BMI'
Step 12: Create a variable named 'sSQL' to store the SQL query "Select * From [Dim-BMI];"
Step 13: Now using query and connection variable we can fetch the data to a data frame name data frame as 'PersonFrame0' using pd.read_sqp_query(sSQL,conn1)
Step 14: Now that we have stored previous data in dataframe create another variable to store another query as
sSQL=" Select PersonID, Height, weight, bmi, Indicator From [Dim-BMI] Where Height >1.5 and Indicator = 1 ORDER BY Height, Weight;" this query will read the above columns with some conditions applied to them from the Dim_BMI table
Step 15: As we have done above now we have a query and a connection variable(conn1) so we can read the data to a dataframe in this step we will do that using the following code 'PersonFrame1 = pd.read_sql_query(sSQL, conn1)'
Step 16: Assign PersonFrame1 to another variable 'DimPerson' then using this dataframe create another dataframe 'DimPersonIndex' with its index set as the 'PersonID' column by this code DimPerson.set_index(['PersonID'],inplace=false)
Step 17: Now we have to store DimPersonIndex dataframe data to a SQL Table using the 'to_sql()' method using this code: DimPersonIndex.to_sql(sTable,conn2,if_exist=" replace")
Step 18: After we have stored horizontally styled data in the SQL table we will retrieve it to another dataframe and compare it with the previous dataframe using
a query "SELECT * FROM [Dim-BMI];" with code PersonFrame2 = pd.read_sql_query(sSQL,conn2)
Step 19: In the last step we will compare PersonFrame0 with PersonFrame2 and observe how many rows and columns we have dropped in horizontal styling.

```
import sys
import os
import pandas as pd
import sqlite3 as sq
print("Ninad Karlekar 22306A1012")
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 ["+sTable+"];"
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 ["+sTable+"];"
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("Ninad Karlekar 22306A1012")
print('#################################')
```

```
In [11]: runfile('F:/MSc IT/Practical/DS/Code files/D
Practical/DS/Code files')
Ninad Karlekar 22306A1012
################################
Working Base : C:/VKHCG  using  win32
################################
###############
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################

################################
Storing : C:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI-Horizontal
Loading : C:/VKHCG/99-DW/datamart.db  T
################################
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 5
Ninad Karlekar 22306A1012
################################
```

Vertical style

Step 1: Import sys, os, pandas as pd, and sqlite3 as sq libraries
Step 2: Create a variable named ‘Base’ and assign it the value ‘C:/VKHCG’
Step 3: Print the working base and platform you are using with the help of the sys.platform
Step 4: Create a variable named ‘Company’ and assign it the value ‘01-Vermeulen’
Step 5: Create a variable to store the warehouse directory and set the value as Base + ‘/99-DW’ . this is the location where you will store your warehouse data
Step 6: Check if the DataWarehouse directory exists or not if it does not exist then create it using the method ‘makedirs’ like os.makedirs(sDataWarehouse)
Step 7: Create a variable to store a database name and assign it to a value as sDataWarehouseDir + ‘/datawarehouse.db’
Step 8: Now create a connection variable ‘conn1’ to store the connection with the datawarehouse database using the sq.connect(databasename_variable) method
Step 9: Create another variable to store the datamart database and assign it a value as sDataWarehouseDir + ‘/datamart.db’
Step 10: Create connection variable conn2 to store connection with the datamart database using the sq.connect() method
Step 11: Create a variable to table name from which we are going to fetch data and assign it ‘Dim-BMI’
Step 12: Create a variable named ‘sSQL’ to store the SQL query “Select * From [Dim-BMI];”
Step 13: Now using query and connection variable we can fetch the data to a data frame name data frame as ‘PersonFrame0’ using pd.read_sqp_query(sSQL,conn1)
Step 14: Now that we have stored previous data in dataframe create another variable to store different query as
sSQL=” Select Height, weight, Indicator From [Dim-BMI];” this query will read the above columns from the Dim_BMI table

Step 15: As we have done above now we have a query and a connection variable(conn1) so we can read the data to a dataframe in this step we will do that using the following code 'PersonFrame1 = pd.read_sql_query(sSQL, conn1)'
Step 16: Assign PersonFrame1 to another variable 'DimPerson' then using this dataframe create another dataframe 'DimPersonIndex' with its index set as the 'Indicator' column by this code DimPerson.set_index(['Indicator'],inplace=false)
Step 17: Now we have to store DimPersonIndex dataframe data to a SQL Table named 'Dim-BMI-Vertical' using the 'to_sql()' method using this code: DimPersonIndex.to_sql(sTable,conn2,if_exist=" replace")
Step 18: After we have stored vertically styled data in the SQL table we will retrieve it to another dataframe and compare it with the previous dataframe using
a query "SELECT * FROM [Dim-BMI-Vertical];" with code PersonFrame2 = pd.read_sql_query(sSQL,conn2)
Step 19: In the last step we will compare PersonFrame0 with PersonFrame2 and observe how many rows and columns we have dropped in vertical styling.

```
import sys
import os
import pandas as pd
import sqlite3 as sq
print("Ninad Karlekar 22306A1012")
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("Ninad Karlekar 22306A1012")
print('#################################')
```

```
Ninad Karlekar 22306A1012
##############################
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
#################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
#################################
Horizontal Data Set (Rows): 1080
Horizontal Data Set (Columns): 3
Ninad Karlekar 22306A1012
#################################
```

Island style

Step 1: Import sys, os, pandas as pd, and sqlite3 as sq libraries
Step 2: Create a variable named ‘Base’ and assign it the value ‘C:/VKHCG’
Step 3: Print the working base and platform you are using with the help of the sys.platform
Step 4: Create a variable named ‘Company’ and assign it the value ‘01-Vermeulen’
Step 5: Create a variable to store the warehouse directory and set the value as Base + ‘/99-DW’. this is the location where you will store your warehouse data
Step 6: Check if the DataWarehouse directory exists or not if it does not exist then create it using the method ‘makedirs’ like os.makedirs(sDataWarehouse)
Step 7: Create a variable to store a database name and assign it to a value as sDataWarehouseDir + ‘/datawarehouse.db’
Step 8: Now create a connection variable ‘conn1’ to store the connection with the datawarehouse database using the sq.connect(databasename_variable) method
Step 9: Create another variable to store the datamart database and assign it a value as sDataWarehouseDir + ‘/datamart.db’
Step 10: Create connection variable conn2 to store connection with the datamart database using the sq.connect() method
Step 11: Create a variable to table name from which we are going to fetch data and assign it ‘Dim-BMI’
Step 12: Create a variable named ‘sSQL’ to store the SQL query “Select * From [Dim- BMI];”
Step 13: Now using query and connection variable we can fetch the data to a data frame name data frame as ‘PersonFrame0’ using pd.read_sqp_query(sSQL,conn1)

Step 14: Now that we have stored previous data in dataframe create another variable to store another query as sSQL="SELECT Height, Weight, Indicator FROM [Dim-BMI] WHERE Indicator > 2 ORDER BY Height, Weight;" this query will read the above columns with some conditions applied to them from the Dim_BMI table
Step 15: As we have done above now we have a query and a connection variable(conn1) so we can read the data to a dataframe in this step we will do that using the following code 'PersonFrame1 = pd.read_sql_query(sSQL, conn1)'
Step 16: Assign PersonFrame1 to another variable 'DimPerson' then using this dataframe create another dataframe 'DimPersonIndex' with its index set as the 'PersonID' column by this code DimPerson.set_index(['PersonID'],inplace=false)
Step 17: Now we have to store DimPersonIndex dataframe data to a SQL Table using the 'to_sql()' method using this code: DimPersonIndex.to_sql(sTable,conn2,if_exist=" replace")
Step 18: After we have stored horizontally styled data in the SQL table we will retrieve it to another dataframe and compare it with the previous dataframe using
a query "SELECT * FROM [Dim-BMI-Vertical];" with code PersonFrame2 = pd.read_sql_query(sSQL,conn2)
Step 19: In the last step we will compare PersonFrame0 with PersonFrame2 and observe how many rows and columns we have dropped in horizontal styling.

```
import sys
import os
import pandas as pd
import sqlite3 as sq
Base='C:/VKHCG'
print("Ninad Karlekar 22306A1012")
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
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")
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#################################')
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
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("Ninad Karlekar 22306A1012")
```

```
Ninad Karlekar 22306A1012
################################
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  Tab
################################
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 771
Horizontal Data Set (Columns): 3
Ninad Karlekar 22306A1012
################################
```

Secure Vault style

Step 1: Import sys, os, pandas as pd, and sqlite3 as sq libraries
Step 2: Create a variable named ‘Base’ and assign it the value ‘C:/VKHCG’
Step 3: Print the working base and platform you are using with the help of the sys.platform
Step 4: Create a variable named ‘Company’ and assign it the value ‘01-Vermeulen’
Step 5: Create a variable to store the warehouse directory and set the value as
Base + ‘/99-DW’. this is the location where you will store your warehouse data
Step 6: Check if the DataWarehouse directory exists or not if it does not exist then create it using the method ‘makedirs’ like os.makedirs(sDataWarehouse)
Step 7:
Create a variable to store a database name and assign it to a value as
sDataWarehouseDir + ‘/datawarehouse.db’
Step 8: Now create a connection variable ‘conn1’ to store the connection with the datawarehouse database using the sq.connect(databasename_variable) method
Step 9: Create another variable to store the datamart database and assign it a value as
sDataWarehouseDir + ‘/datamart.db’

Step 10: Create connection variable conn2 to store connection with the datamart database using the sq.connect() method
Step 11: Create a variable to table name from which we are going to fetch data and assign it 'Dim-BMI'
Step 12: Create a variable named 'sSQL' to store the SQL query "Select * From [Dim- BMI];"
Step 13: Now using query and connection variable we can fetch the data to a data frame name data frame as 'PersonFrame0' using pd.read_sqp_query(sSQL,conn1)
Step 14: Now that we have stored previous data in dataframe create another variable to store another query as 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;" this query will read the above columns with some conditions applied to them from the Dim_BMI table
Step 15:As we have done above now we have a query and a connection variable(conn1) so we can read the data to a dataframe in this step we will do that using the following code 'PersonFrame1 = pd.read_sql_query(sSQL, conn1)'
Step 16: Assign PersonFrame1 to another variable 'DimPerson' then using this dataframe create another dataframe 'DimPersonIndex' with its index set as the 'PersonID' column by this code DimPerson.set_index(['PersonID'],inplace=false)
Step 17: Now we have to store DimPersonIndex dataframe data to a SQL Table using the 'to_sql()' method using this code: DimPersonIndex.to_sql(sTable,conn2,if_exist=" replace")
Step 18: After we have stored horizontally styled data in the SQL table we will retrieve it to another dataframe and compare it with the previous dataframe using a query "SELECT * FROM [Dim-BMI-Horizontall];" with code PersonFrame2 = pd.read_sql_query(sSQL,conn2)
Step 19: In the last step we will compare PersonFrame0 with PersonFrame2 and observe how many rows and columns we have dropped in horizontal styling.
Step 20: PersonFrame2.head() .Return the first 5 rows.

```
import sys
import os
import pandas as pd
import sqlite3 as sq
print("Ninad Karlekar 22306A1012")
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('Storing :',sDatabaseName,'\n Table:',sTable)
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI-Secure'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('Only Sam Data')
print(PersonFrame2.head())
print("Ninad Karlekar 22306A1012")
```

```
Practical/DS/Code Files )
Ninad Karlekar 22306A1012
################################
Working Base : C:/VKHCG  using  win32
################################
################
Loading : C:/VKHCG/99-DW/datamart.db  T
################
Loading : C:/VKHCG/99-DW/datamart.db  T

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

################################
################################
Loading : C:/VKHCG/99-DW/datamart.db  T
################################
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
################################
################################
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
Ninad Karlekar 22306A1012
################################
```

VSIT

DATA SCIENCE
Practical # 7

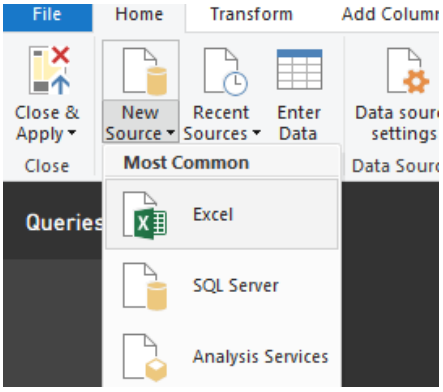
Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Data Visualization with Power BI	**Division**	A

Case Study: Sales Data

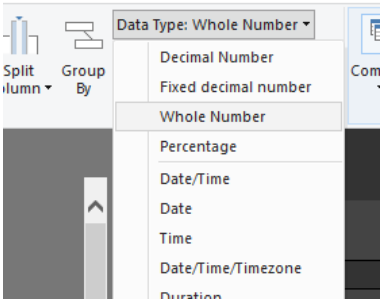
Case Study: Sales Data

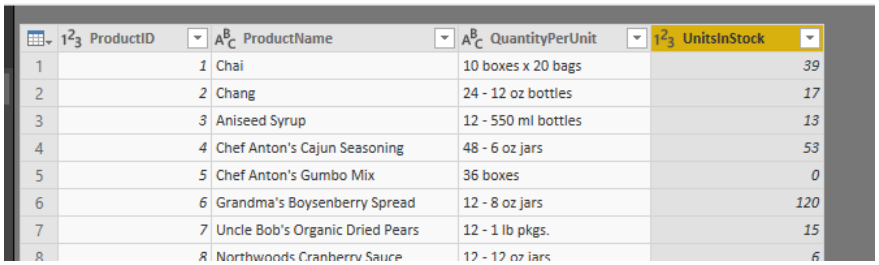
Step 1: Connect to an Excel workbook

1. Launch power BI Desktop
2. Home -> Get Data -> Excel -> open product.xlsx
3. Right click on product -> Edit



4. In Query Editor, select the ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns (use Ctrl + Click to select more than one column, or Shift + Click to select columns that are beside each other)
5. Select Remove Columns -> Remove Other Columns from the ribbon, or right-click on a column header and click Remove Other Columns.
6. Change the data type of the UnitsInStock column
   a. Select the UnitsInStock column.
   b. Select the Data Type drop-down button in the Home ribbon.
   c. If not already a Whole Number, select Whole Number for data type from the drop down *(the Data Type: button also displays the data type for the current selection).*





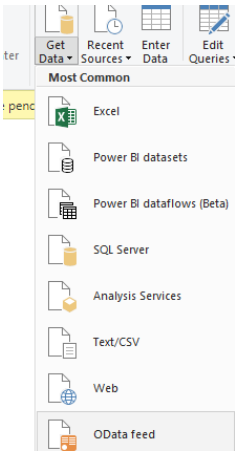
	ProductID	ProductName	QuantityPerUnit	UnitsInStock
1	1	Chai	10 boxes x 20 bags	39
2	2	Chang	24 - 12 oz bottles	17
3	3	Aniseed Syrup	12 - 550 ml bottles	13
4	4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	53
5	5	Chef Anton's Gumbo Mix	36 boxes	0
6	6	Grandma's Boysenberry Spread	12 - 8 oz jars	120
7	7	Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	15
8	8	Northwoods Cranberry Sauce	12 - 12 oz jars	6

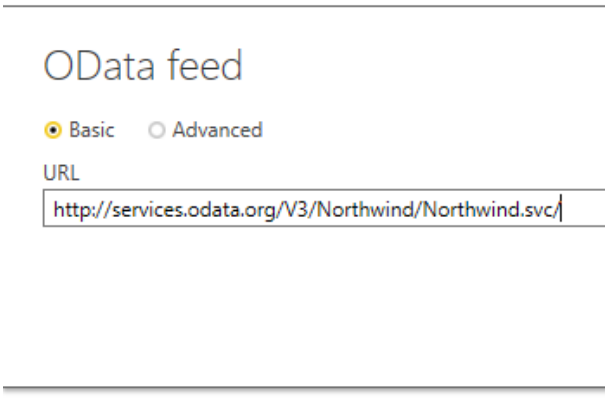
Task 2: Import order data from an OData feed

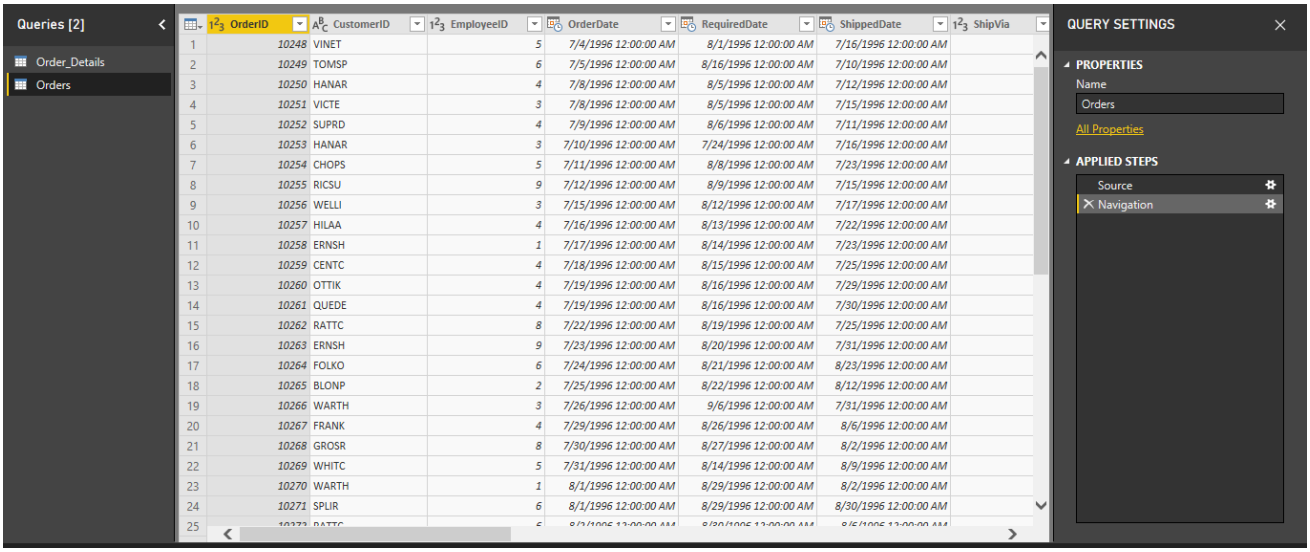
You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below:
http://services.odata.org/V3/Northwind/Northwind.svc/

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.



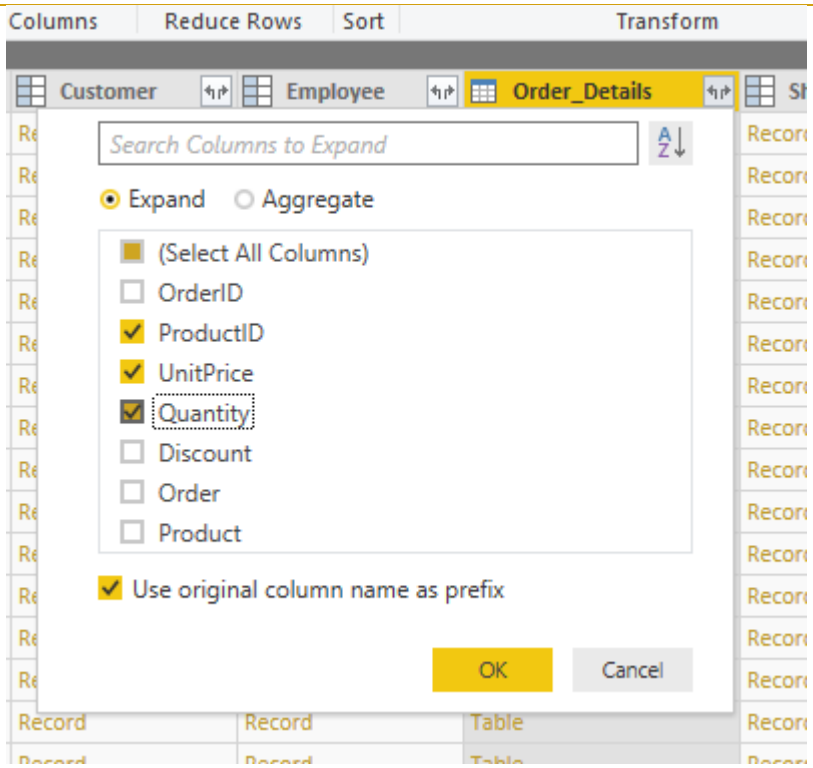




	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia
1	10248	VINET	5	7/4/1996 12:00:00 AM	8/1/1996 12:00:00 AM	7/16/1996 12:00:00 AM	
2	10249	TOMSP	6	7/5/1996 12:00:00 AM	8/16/1996 12:00:00 AM	7/10/1996 12:00:00 AM	
3	10250	HANAR	4	7/8/1996 12:00:00 AM	8/5/1996 12:00:00 AM	7/12/1996 12:00:00 AM	
4	10251	VICTE	3	7/8/1996 12:00:00 AM	8/5/1996 12:00:00 AM	7/15/1996 12:00:00 AM	
5	10252	SUPRD	4	7/9/1996 12:00:00 AM	8/6/1996 12:00:00 AM	7/11/1996 12:00:00 AM	
6	10253	HANAR	3	7/10/1996 12:00:00 AM	7/24/1996 12:00:00 AM	7/16/1996 12:00:00 AM	
7	10254	CHOPS	5	7/11/1996 12:00:00 AM	8/8/1996 12:00:00 AM	7/23/1996 12:00:00 AM	
8	10255	RICSU	9	7/12/1996 12:00:00 AM	8/9/1996 12:00:00 AM	7/15/1996 12:00:00 AM	
9	10256	WELLI	3	7/15/1996 12:00:00 AM	8/12/1996 12:00:00 AM	7/17/1996 12:00:00 AM	
10	10257	HILAA	4	7/16/1996 12:00:00 AM	8/13/1996 12:00:00 AM	7/22/1996 12:00:00 AM	
11	10258	ERNSH	1	7/17/1996 12:00:00 AM	8/14/1996 12:00:00 AM	7/23/1996 12:00:00 AM	
12	10259	CENTC	4	7/18/1996 12:00:00 AM	8/15/1996 12:00:00 AM	7/25/1996 12:00:00 AM	
13	10260	OTTIK	4	7/19/1996 12:00:00 AM	8/16/1996 12:00:00 AM	7/29/1996 12:00:00 AM	
14	10261	QUEDE	4	7/19/1996 12:00:00 AM	8/16/1996 12:00:00 AM	7/30/1996 12:00:00 AM	
15	10262	RATTC	8	7/22/1996 12:00:00 AM	8/19/1996 12:00:00 AM	7/25/1996 12:00:00 AM	
16	10263	ERNSH	9	7/23/1996 12:00:00 AM	8/20/1996 12:00:00 AM	7/31/1996 12:00:00 AM	
17	10264	FOLKO	6	7/24/1996 12:00:00 AM	8/21/1996 12:00:00 AM	8/23/1996 12:00:00 AM	
18	10265	BLONP	2	7/25/1996 12:00:00 AM	8/22/1996 12:00:00 AM	8/12/1996 12:00:00 AM	
19	10266	WARTH	3	7/26/1996 12:00:00 AM	9/6/1996 12:00:00 AM	7/31/1996 12:00:00 AM	
20	10267	FRANK	4	7/29/1996 12:00:00 AM	8/26/1996 12:00:00 AM	8/6/1996 12:00:00 AM	
21	10268	GROSR	8	7/30/1996 12:00:00 AM	8/27/1996 12:00:00 AM	8/2/1996 12:00:00 AM	
22	10269	WHITC	5	7/31/1996 12:00:00 AM	8/14/1996 12:00:00 AM	8/9/1996 12:00:00 AM	
23	10270	WARTH	1	8/1/1996 12:00:00 AM	8/29/1996 12:00:00 AM	8/2/1996 12:00:00 AM	
24	10271	SPLIR	6	8/1/1996 12:00:00 AM	8/29/1996 12:00:00 AM	8/30/1996 12:00:00 AM	
25							

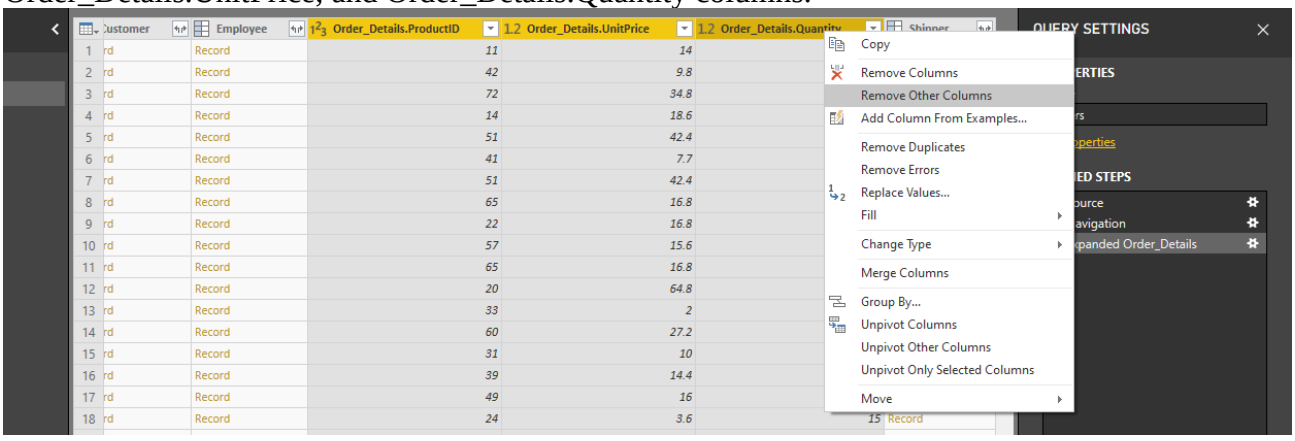
Step 2: Expand the Order_Details table

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.
   b. Select ProductID, UnitPrice, and Quantity.
   c. click OK.



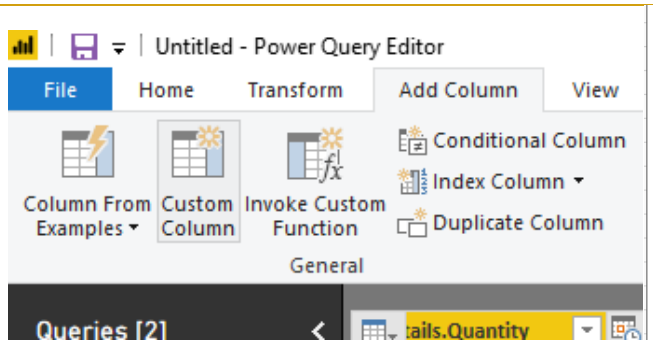
Step 3: Remove other columns to only display columns of interest

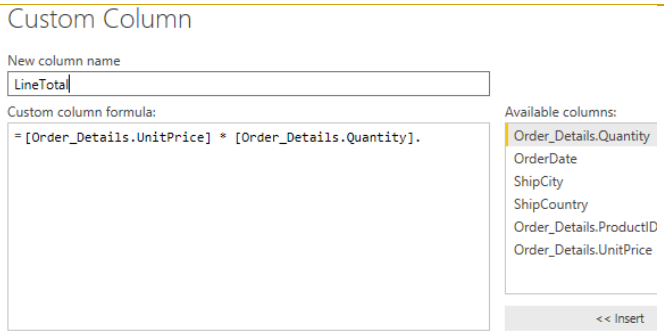
In this step you remove all columns except OrderDate, ShipCity, ShipCountry, In this step you remove all columns except OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and Order_Details.Quantity columns.



Step 4: Calculate the line total for each Order_Details row

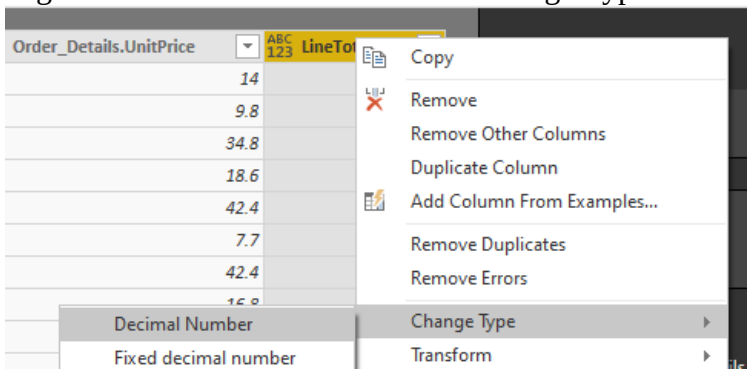
1. In the Add Column ribbon tab -> Add Custom Column.
2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order_Details.UnitPrice] * [Order_Details.Quantity]
3. In the New column name textbox, enter LineTotal.





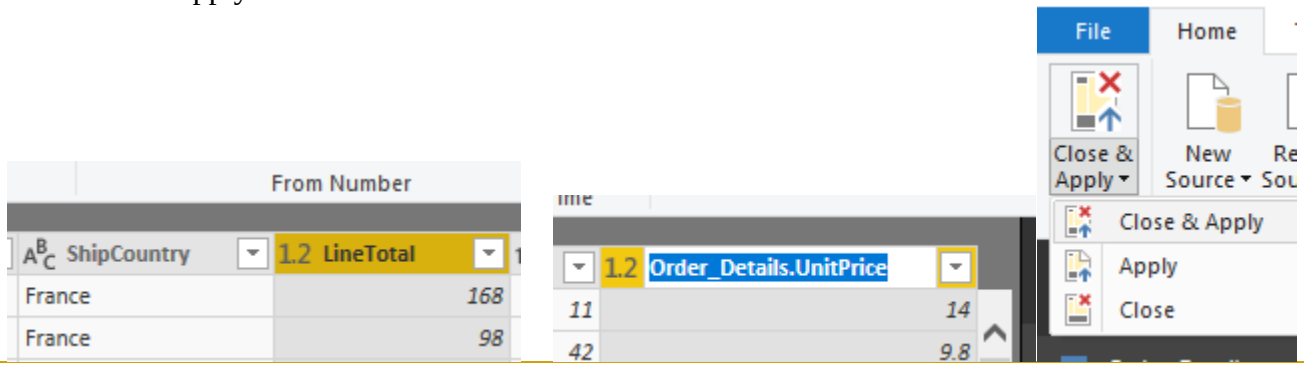
Step 5: Set the datatype of the LineTotal field

Right click the LineTotal column -> Change Type -> Decimal Number



Step 6: Rename and reorder columns in the query

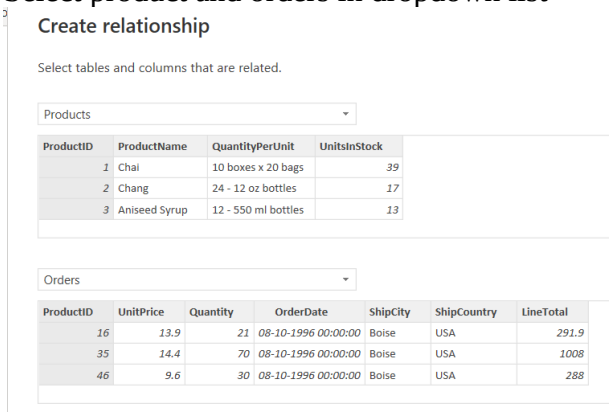
1. In Query Editor, drag the LineTotal column to the left, after ShipCountry.
2. Remove the Order_Details. prefix from the Order_Details.ProductID, Order_Details.UnitPrice and Order_Details.Quantity columns, by double-clicking on each column header, and then deleting that text from the column name.
3. Close & Apply.



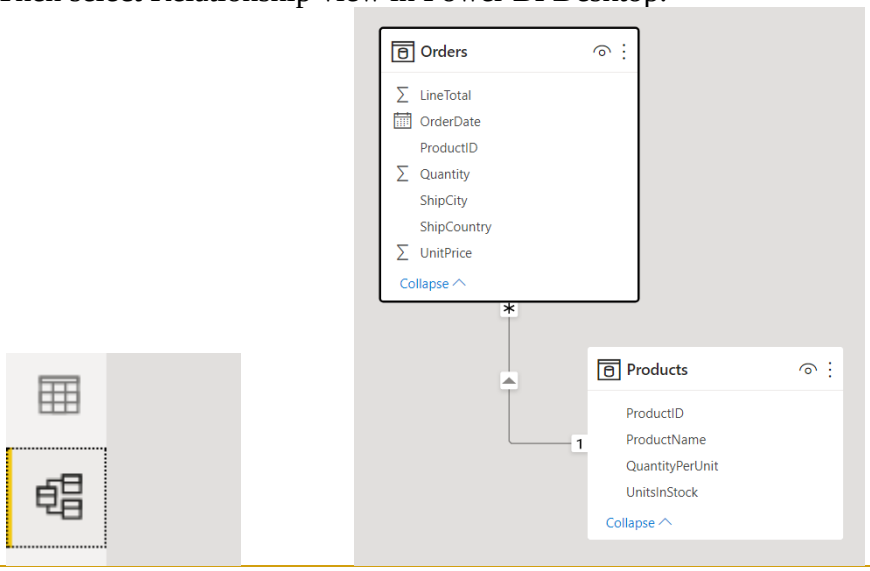
Task 3: Combine the Products and Total Sales queries

1. Home tab -> Manage Relationships -> New

2. Select product and orders in dropdown list

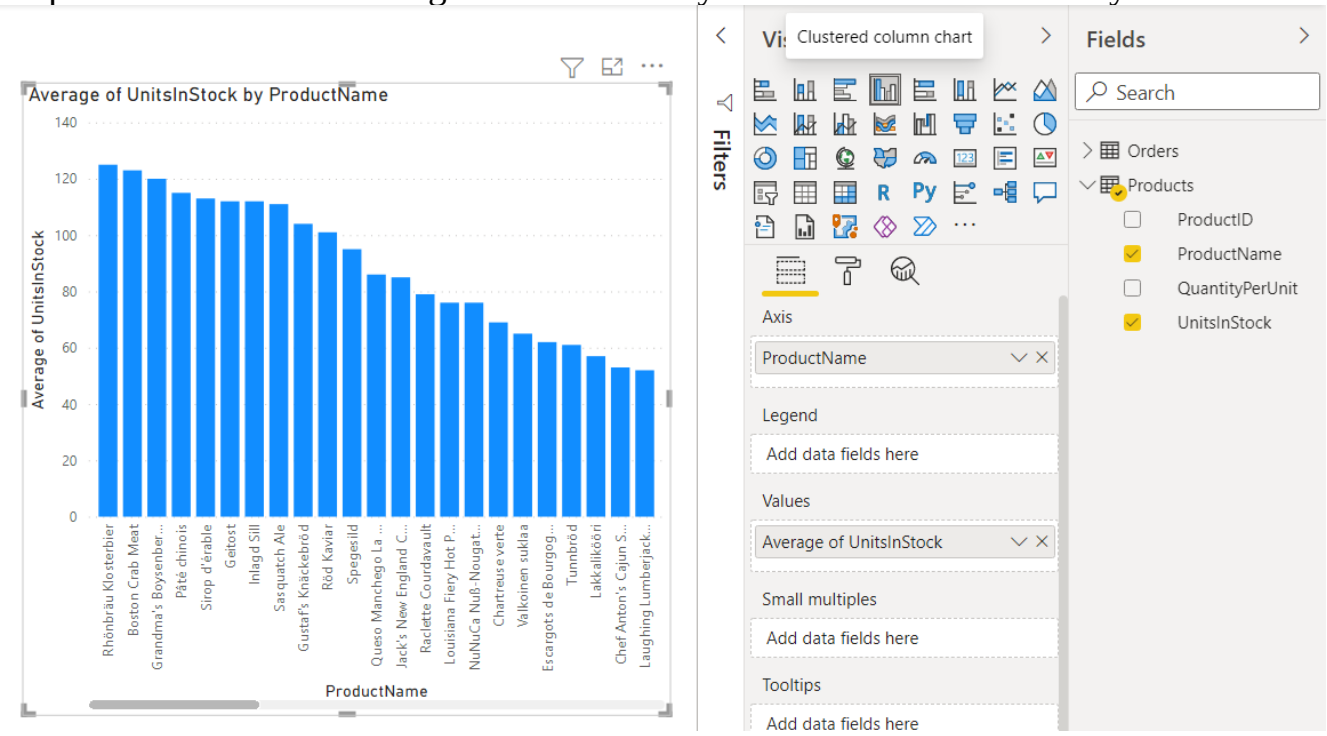


3. Then select Relationship view in Power BI Desktop.

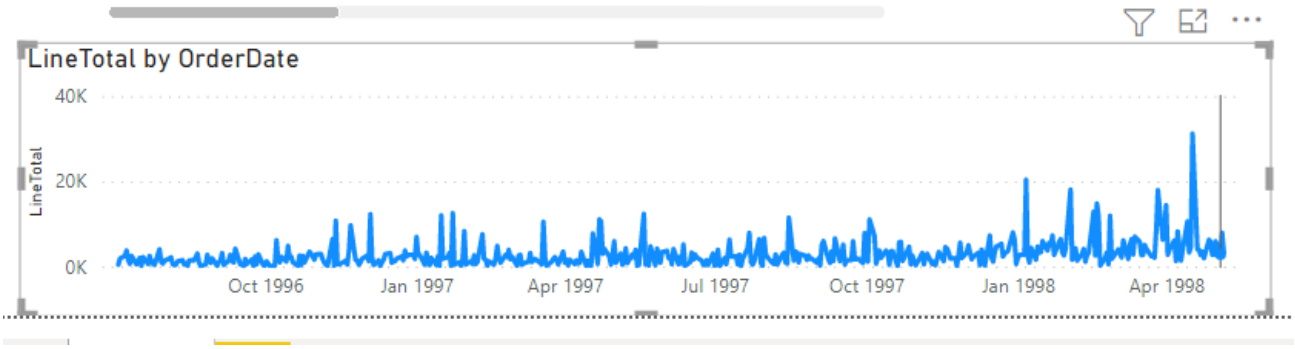


Task 4: Build visuals using your data

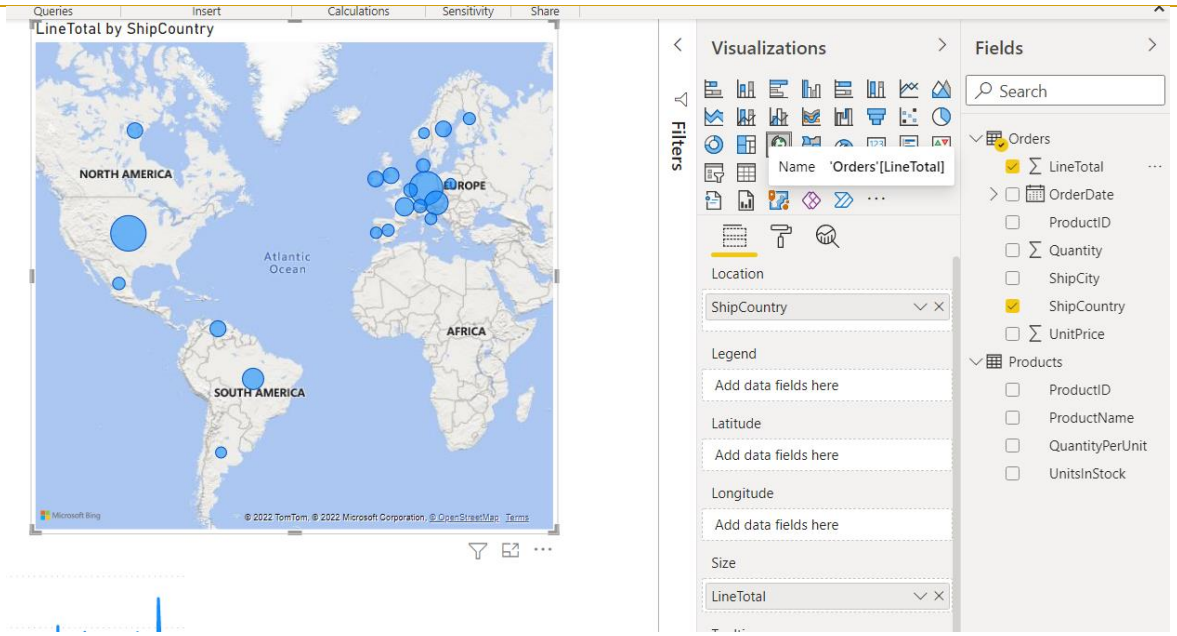
Step 1: Create charts showing Units in Stock by Product and Total Sales by Year



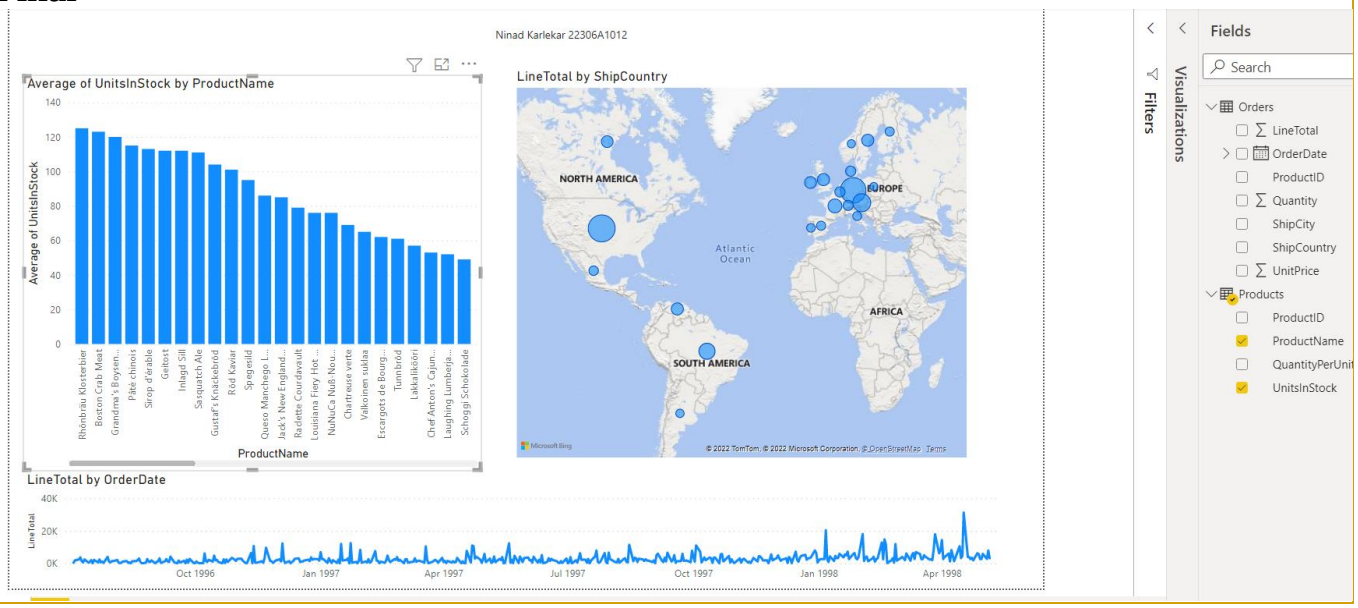
Step 2: Drag OrderDate to the canvas beneath the first chart, then drag LineTotaI (again, from the Fields pane) onto the visual, then select Line Chart. The following visualization is created.



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



Step 4: Interact with your report visuals to analyse further Final



VSIT

DATA SCIENCE
Practical # 8

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Transforming Data	**Division**	A

Building data warehouse

Steps:
1.Import the following libraries pandas as pd, and sqlite3 as sq libraries
Sys, os, uuid (if not installed install them through pip install 'library-name')

2.Create variable base and assign it the value of your VKHCG directory (need to perform transformation before building a data warehouse)

3.Define or create variables such as 'sDataBaseDir', 'sDatabaseName', 'sDataWarehousedir' to store a database name

4.Check if the directory exists or not if not, you can create it by using 'makedirs' method

5.Create connection variables such as conn1, conn2, conn3 for

'sDatabaseName', 'sDataVaultDir', 'sDataWarehouseDir'

6.Store sql queries inside sSQL variable

7.Fetch values from Hub-Time-Gunnarsson table using sql query

8.Converting and printing 'DateTimeValue' into python datetime object using strptime method

9.Read the data into a dataframe

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

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

```
################################
>>>
 RESTART: C:\Users\viggu\Desktop\after\VKHCG\01-Vermeulen\04-Transform\Transform-Sun-Models.py
################################
Working Base : C:/VKHCG  using  win32
################################
                      DateTimeValue
0  1960-12-20 10:15:00 (GMT) (+0000)

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

################################
1 3
2 3
3 3

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

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

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

################################
1 1
                                PersonID  ... Zone
0  e780efb2-e4e6-4f85-9d33-4368f2308790  ...  UTC

[1 rows x 4 columns]

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

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

Simple Linear Regression

Steps:

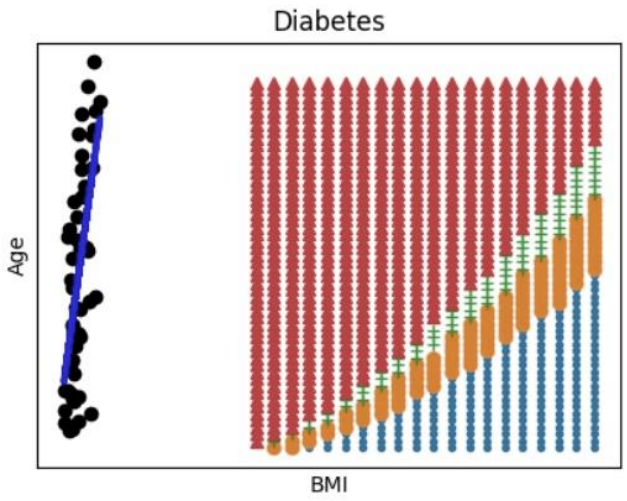
1. In the Python editor, open a new file named simple_regression.py
2. Save it in directory (C:\VKHCG\01-Vermeulen\04-Transform)
3. Import the libraries matplotlib, pandas as pd, sqlite as sq, matplotlib as plt, datasets from sklearn which are linear_modell and mean_squared_error, r2_score
4. Create variable base and assign it the path 'C:/VKHCG'
5. Check if present or create the database directory, database vault directory, Datawarehouse directory variables along with base path using makedirs method
6. Connect to database using conn1, conn2, conn3 variables through sq
7. Create tMax variable
8. Using for loop define the range for height and weight
9. Create bmi variable and assign formula
10. Compare bmi results through if elif statements
11. Read the data into dataframes
12. Plot height and weight using x and y variables
13. Use plt.plot function to display graph
14. Load the diabetes dataset using diabetes = datasets.load_diabetes()
15. Split the dataset into X_train, X_test, y_train, y_test and fetch values using slice operator
16. Use variable regr to call the linear_model.LinearRegression() function.
17. Use fit method on regr to pass the two
18. Use plt.show to plot the graph

```
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()
```

```
Row: 1080 of 1080.0

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

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

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

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

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

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



VSIT

DATA SCIENCE
Practical # 9

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Generating Reports	**Batch**	1

A. Vermeulen PLC

Steps:
Step 1) We need to import all the necessary packages (If u don't have the necessary packages install it using pip)

Step 2) Now we need to store data from the csv file using read_csv function.
Then we need to load the other necessary files.

Step 3) Now we'll create a for loop to build our program logic.
This for loop will help us to built data links of all the customers.

Step 4) After that store the output file using write_gml function.

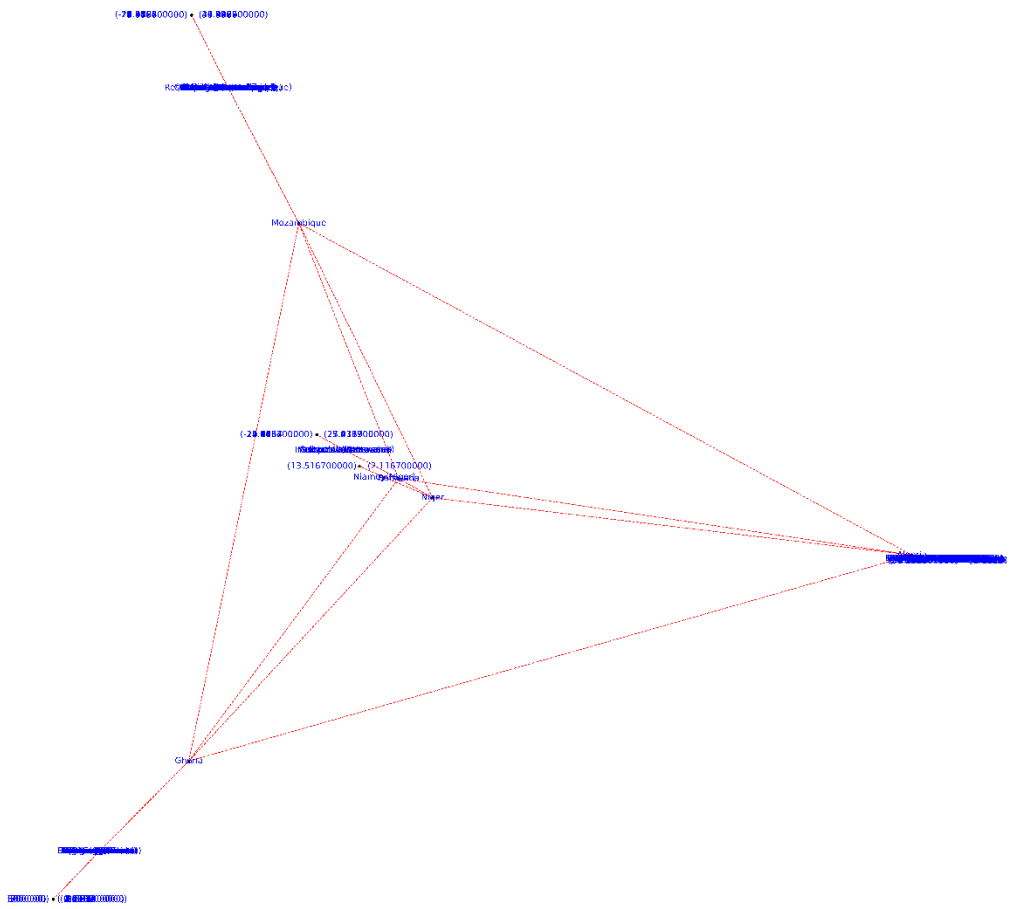
Step 5) At last we will plot a graph/figure of our dataset according to given data links.

```
#######################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
#######################################################################
pd.options.mode.chained_assignment = None
print("Ninad Karlekar 22306A1012")
#######################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
#######################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#######################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv'
#######################################################################
sOutputFileName1='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml'
```

```
sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png'
Company='01-Vermeulen'
#################################################################
#################################################################
### Import Country Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
CustomerDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
CustomerData=CustomerDataRaw.head(100)
print('Loaded Country:',CustomerData.columns.values)
print('#################################')
#################################################################
print(CustomerData.head())
print(CustomerData.shape)
#################################################################
G=nx.Graph()
for i in range(CustomerData.shape[0]):
    for j in range(CustomerData.shape[0]):
        Node0=CustomerData['Customer_Country_Name'][i]
        Node1=CustomerData['Customer_Country_Name'][j]
        if Node0 != Node1:
            G.add_edge(Node0,Node1)

for i in range(CustomerData.shape[0]):
    Node0=CustomerData['Customer_Country_Name'][i]
    Node1=CustomerData['Customer_Place_Name'][i] + '('+
CustomerData['Customer_Country_Name'][i] + ')'
    Node2='('+ "{:.9f}".format(CustomerData['Customer_Latitude'][i]) + ')\
    ('+ "{:.9f}".format(CustomerData['Customer_Longitude'][i]) + ')'
    if Node0 != Node1:
        G.add_edge(Node0,Node1)
    if Node1 != Node2:
        G.add_edge(Node1,Node2)

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

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

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

print('##################################')
print("Ninad Karlekar 22306A1012")
print('### Done!! ####################')
print('##################################')
#####################################################################
```

```
In [18]: runfile('C:/Users/User/a/untitled5.py', wdir='C:/Users/User/a')
Ninad Karlekar 22306A1012
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-
Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
################################
  Customer_Country_Code  ... Customer_Country_Name
0                    BW  ...              Botswana
1                    BW  ...              Botswana
2                    BW  ...              Botswana
3                    BW  ...              Botswana
4                    NE  ...                 Niger

[5 rows x 5 columns]
(100, 5)
Nodes: 205
Edges: 210
################################
Storing : C:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Network-Routing-
Customer.gml
################################
################################
Storing Graph Image: C:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Network-
Routing-Customer.png
################################
################################
Ninad Karlekar 22306A1012
### Done!! ####################
################################
```



B. Krennwallner AG

Steps:
Step 1) We need to import all the necessary packages
(If u don't have the necessary packages install it using pip)

Step 2) Now we need to store data from the csv file using read_csv function. Then we need to load the other necessary files.

Step 3) We will create a for loop to build longitude and latitude of the locations and description.

Step 4) To fullfil null values we assign them with 0.

Step 5) Now we will store longitude latitude and description in list.

Step 6) After that we will display our data on map using "Map function"

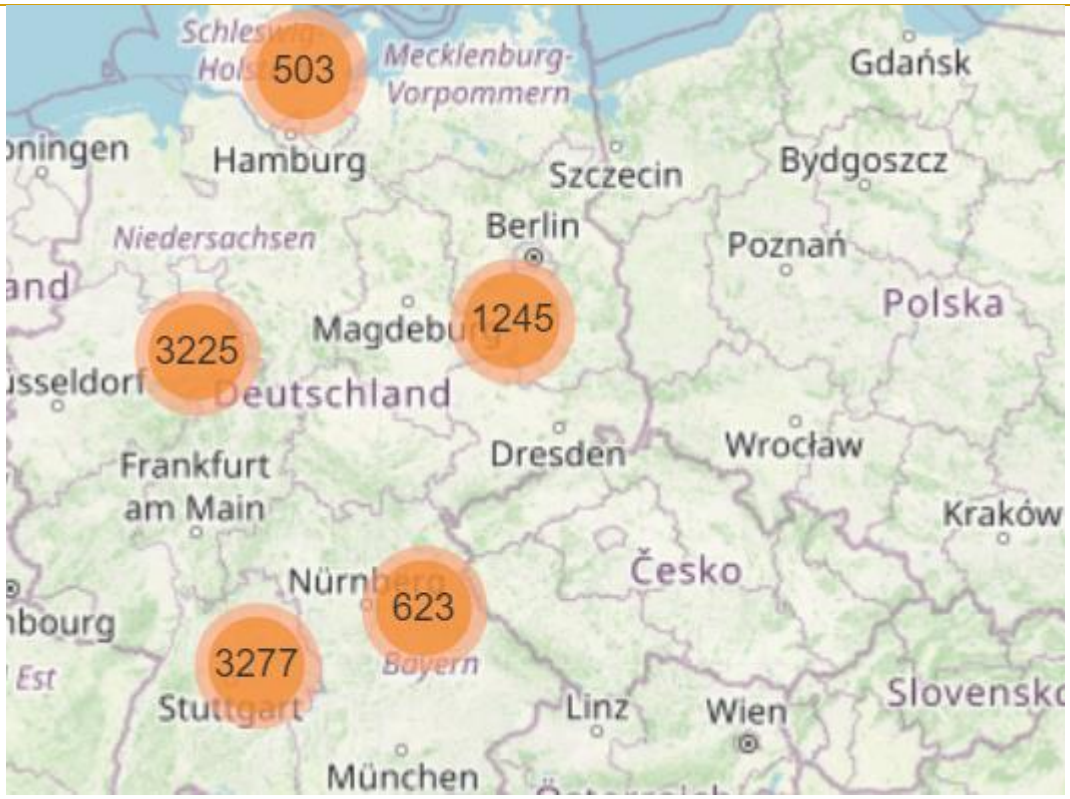
Step 7) We will store 3 different types of maps which will display geographical location of billboard

Step 8) We save the output in html file

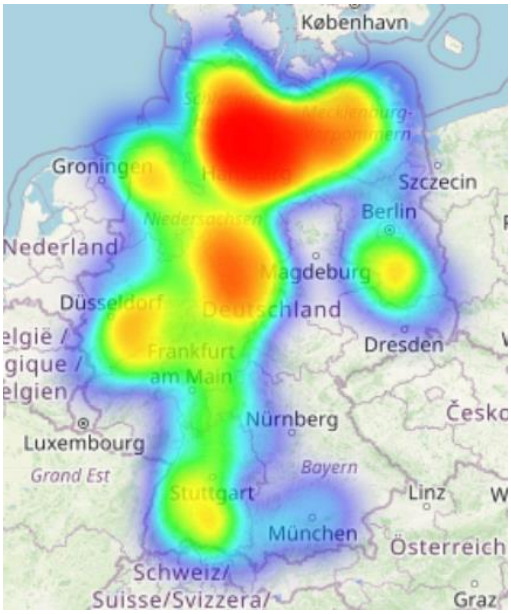
```
import sys
import os
import pandas as pd
from folium.plugins import FastMarkerCluster, HeatMap
from folium import Marker, Map
import webbrowser
print("Ninad Karlekar 22306A1012")

#####################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sFileName=Base+'/02-Krennwallner/01-Retrieve/01-EDS/02-
Python/Retrieve_DE_Billboard_Locations.csv'
df = pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
df.fillna(value=0, inplace=True)
print(df.shape)
t=0
for i in range(df.shape[0]):
    try:
        sLongitude=df["Longitude"][i]
        sLongitude=float(sLongitude)
    except Exception:
        sLongitude=float(0.0)

    try:
        sLatitude=df["Latitude"][i]
        sLatitude=float(sLatitude)
    except Exception:
        sLatitude=float(0.0)

    try:
        sDescription=df["Place_Name"][i] + ' (' + df["Country"][i]+')'
    except Exception:
        sDescription='VKHCG'

    if sLongitude != 0.0 and sLatitude != 0.0:
        DataClusterList=list([sLatitude, sLongitude])
        DataPointList=list([sLatitude, sLongitude, sDescription])
        t+=1
        if t==1:
```

```
            DataCluster=[DataClusterList]
            DataPoint=[DataPointList]
        else:
            DataCluster.append(DataClusterList)
            DataPoint.append(DataPointList)
data=DataCluster
pins=pd.DataFrame(DataPoint)
pins.columns = [ 'Latitude','Longitude','Description']
stops_map1 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
marker_cluster = FastMarkerCluster(data).add_to(stops_map1)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard1.html'
stops_map1.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
stops_map2 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
for name, row in pins.iloc[:100].iterrows():
    Marker([row["Latitude"],row["Longitude"]], popup=row["Description"]).add_to(stops_map2)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html'
stops_map2.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
stops_heatmap = Map(location=[48.1459806, 11.4985484], zoom_start=5)
stops_heatmap.add_child(HeatMap([[row["Latitude"], row["Longitude"]] for name, row in
pins.iloc[:100].iterrows()]))
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard_heatmap.html'
stops_heatmap.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
print('### Done!! ############################################')
print("Ninad Karlekar 22306A1012")
```

```
In [3]: runfile('F:/MSC IT/Practical/DS/Code files/DS_Prac
Practical/DS/Code files')
Ninad Karlekar 22306A1012
################################
Working Base : C:/VKHCG  using  win32
################################
(8873, 4)
### Done!! ############################################
Ninad Karlekar 22306A1012

In [4]:
```


503
1245
3225
623
3277
Gdańsk
Hamburg
Szczecin
Bydgoszcz
Berlin
Poznań
Polska
Deutschland
Dresden
Wrocław
Frankfurt
am Main
Kraków
Česko
Linz
Wien
München


København
Luxembourg
Dresden
Wrocław
Česko
Linz
Wien
München


København
Szczecin
Berlin
Magdeburg
Dresden
Nürnberg
Luxembourg
Linz
München

VSIT

DATA SCIENCE
Practical # 10

Name	Ninad Karlekar	**Roll Number**	22306A1012
Subject/Course:	DATA SCIENCE	**Class**	M.Sc. IT – Sem I
Topic	Processing Data	**Batch**	1

A. Build the time Hub, Link and Satellite

1. Go to google and search 'VKHCG GitHub' -> Download the zip file -> cut and paste VKHCG folder to 'C-Drive'
2. Open your Python editor and create a file named Process_Time.py.
3. Save it into directory C:\VKHCG\01-Vermeulen\03-Process.
4. Write the code in Process_Time.py file and run.
5. The database has been created in following directory (...\ VKHCG\88-DV\datavault.db.)

```
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)
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
base = datetime(2018,1,1,0,0,0)
numUnits=10*365*24
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)]
```

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

```
            TimeZoneFrame = pd.DataFrame.from_items(TimeZoneLine)
        else:
            TimeZoneRow = pd.DataFrame.from_items(TimeZoneLine)
            TimeZoneFrame = TimeZoneFrame.append(TimeZoneRow)

    TimeZoneFrameIndex=TimeZoneFrame.set_index(['IDZoneNumber'],inplace=False)
    sZone=zone.replace('/','-').replace(' ','')

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

```
2016-12-05 02:00:00  2015-10-07 11:00:00
2016-12-05 01:00:00  2015-10-07 10:00:00
2016-12-05 00:00:00  2015-10-07 09:00:00
2016-12-04 23:00:00  2015-10-07 08:00:00
2016-12-04 22:00:00  2015-10-07 07:00:00
2016-12-04 21:00:00  2015-10-07 06:00:00
2016-12-04 20:00:00  2015-10-07 05:00:00
2016-12-04 19:00:00  2015-10-07 04:00:00
2016-12-04 18:00:00  2015-10-07 03:00:00
2016-12-04 17:00:00  2015-10-07 02:00:00
2016-12-04 16:00:00  2015-10-07 01:00:00
2016-12-04 15:00:00  2015-10-07 00:00:00
```

B. Human-Environment Interaction

1. Go to google and search ‘VKHCG GitHub’ -> Download the zip file -> cut and paste VKHCG folder to ‘C-Drive’
2. In the Python editor, open a new file named Process_Location.py
3. Save it in directory (...\VKHCG\01-Vermeulen\03-Process.)
4. Write the code in Process_ Location.py file and run.

```
import sys
import os
import pandas as pd
import sqlite3 as sq
```

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

```
  LocationFrame = LocationFrame.append(LocationRow)
Create: 637  of  64800 : L+170000--030000
Create: 638  of  64800 : L+170000--020000
Create: 639  of  64800 : L+170000--010000
Create: 640  of  64800 : L+170000-+000000
Create: 641  of  64800 : L+170000-+010000
Create: 642  of  64800 : L+170000-+020000
Create: 643  of  64800 : L+170000-+030000
Create: 644  of  64800 : L+170000-+040000
Create: 645  of  64800 : L+170000-+050000
Create: 646  of  64800 : L+170000-+060000
Create: 647  of  64800 : L+170000-+070000
Create: 648  of  64800 : L+170000-+080000
c:\vkhcg\01-vermeulen\03-process\process_location.py:
```

C. Forecasting

1. Go to google and search ‘VKHCG GitHub’ -> Download the zip file -> cut and paste VKHCG folder to ‘C-Drive’
2. Open a new file in your Python editor as Process-Shares-Data.py
3. Save it in directory (C: \VKHCG\04-Clark\03-Process)
4. Type pip install quandl in cmd.
5. Write the code in Process-Shares-Data.py file and run.

```
import sys
import os
import sqlite3 as sq
import quandl
import pandas as pd
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
Company='04-Clark'
sInputFileName='00-RawData/VKHCG_Shares.csv'
sOutputFileName='Shares.csv'
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
sFileDir1=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
```

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

```
    sTable=str(RawData['sTable'][sShare])
    print('Storing :',sDatabaseName,' Table:',sTable)
    ShareData.to_sql(sTable, conn, if_exists="replace")
    print('################')
    ###################################################################
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
    print('################################')
    ###################################################################
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir2 + '/Assess_' + sOutputFileName
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir3 + '/Process_' + sOutputFileName
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
```

```
Ninad Karlekar 22306A1012
##############################
Working Base : C:/VKHCG  using  win32
##############################
##############################
Loading : C:/VKHCG/04-Clark/00-RawData/VKHCG_Shares.csv
##############################
Rows    : 10
Columns: 3
###############
##############################
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_Shares.csv
##############################
##############################
##############################
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_Shares.csv
##############################
##############################
##############################
##################################
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_FED_RXI_N_A_CA.csv
##################################
##################################
##################################
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_FED_RXI_N_A_CA.csv
##################################
##################################
Ninad Karlekar 22306A1012
### Done!! ############################################

In [10]:
```