DEPARTMENT OF INFORMATION TECHNOLOGY & DATA SCIENCE

VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY

Practical Journal

Submitted in partial fulfilment of the

requirements for the award of the Degree of

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

Semester – I

By

SANJANA HEMANT MOODLIAR

Seat No.: _____

Paper – 1: DATA SCIENCE
Paper – 2: SOFT COMPUTING TECHNIQUES

DEPARTMENT OF INFORMATION TECHNOLOGY

VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY

(Affiliated to University of Mumbai)

MUMBAI, 400037, MAHARASHTRA

Academic Year 2024 – 2025

VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY

Practical Journal

SANJANA HEMANT MOODLIAR

Seat No.: _____

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

Semester – I

2024 – 2025

Paper – 1: DATA SCIENCE

VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY

Vidyalankar School of Information Technology

Wadala (E), Mumbai 400037

Affiliated to Mumbai University

CERTIFICATE

This is to certify that Miss. **SANJANA HEMANT MOODLIAR** Seat No.: _____of **M. Sc.IT Part 1 – Semester 1** has completed the practical work in the subject of **DATA SCIENCE** during the academic year 2024 – 2025 under the guidance of **Prof. Pushpa Mahapatro** being the partial requirement for the fulfilment of the curriculum of Degree of Master of Science in Information Technology, University of Mumbai.

Place: VSIT, Wadala

Date:

Subject In-Charge	Co-Ordinator
Prof. Pushpa Mahapatro	Dr. Ujwala Sav

Principal

Internal Examiner	College Seal	External Examiner

Vidyalankar School of Information Technology
RE-ACCREDITED : GRADE 'A' BY NAAC

DATA SCIENCE

PRACTICAL SUBJECT CODE:

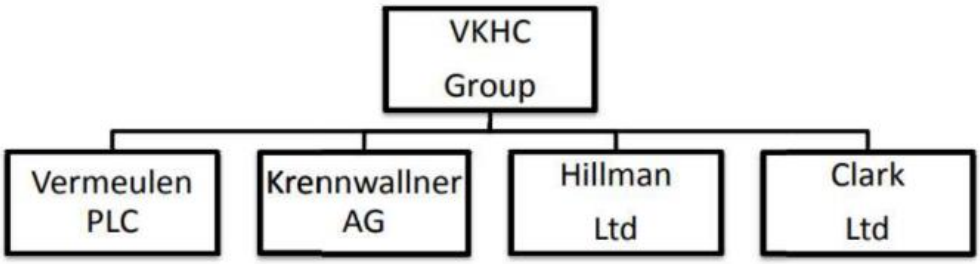
Name: - SANJANA MOODLIAR **Roll no: - 24306A1027**

Practical NO: - 01

Aim:- Overview of Practical and Installation.

Vermeulen-Krennwallner-Hillman-Clark Group (VKHCG) is a hypothetical medium-size international company. It consists of four subcompanies: Vermeulen PLC, Krennwallner AG, Hillman Ltd, and Clark Ltd.

Software requirements:

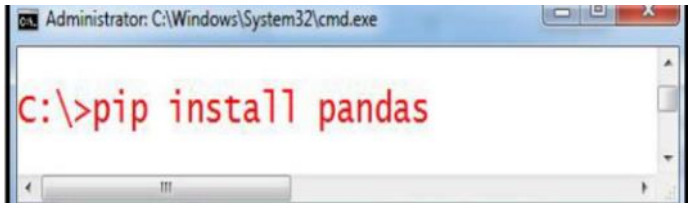


Vermeulen PLC is a data processing company that processes all the data within thegroup companies. The company handles allthe information technology aspects of the business.	**Krennwallner AG** is an advertising and media company that prepares advertising andmedia content for the customers of the group.	**The Hillman** company is a supply chain and logistics company.	**The Clark** company is a venture capitalist and accounting company .
The company supplies • Data science • Networks, servers, and communication systems • Internal and external web sites • Data analysis business activities • Decision science • Process automation • Management reporting	*The company supplies* • Advertising on billboards • Advertising and content management for online delivery • Event management for key customers	*The Company provisions* a worldwide supply chain solution to the businesses, including • Third-party warehousing • International shipping • Door-to-door logistics	*The Company processes* the following financial responsibilities of the group: • Financial insights • Venture capital management • Investments planning • Forex (foreign exchange) trading

- R-Console 3.XXX or Above
- R Studio 1.XXX or above
- Python 2.7 for Cassandra and 3.XXX or above

o While installing Python check the option to Add Python to PATH Variable

o Open CMD in Administrative Mode



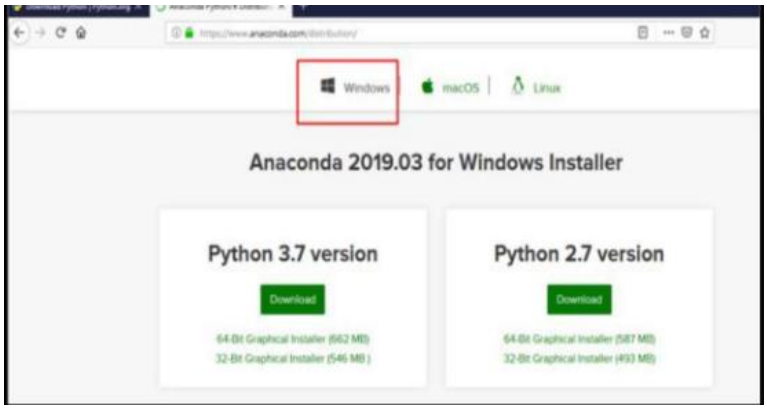
o Similarly install the following packages using pip

1.	matplotlib	8.	datetime	15.	sqlalchemy
2.	numpy	9.	json	16.	sql.connector
3.	opencv-python	10.	msgpack	17.	geopandas
4.	networkx	11.	scipy	18.	quandl
5.	sys	12.	geopy	19.	mlxtend
6.	uuid	13.	pysqlite3	20.	folium
7.	pyspark	14.	openpyxl		

Packages can also be installed using Anaconda

Download Anaconda from https://www.anaconda.com and visit downloads tab.

In the downloads page, scroll down until you see the download options for windows. Click on the download button for python 3.7. This will initiate a download for the anaconda installer.



Follow through the instructions for installing as shown in the next few images. Choose any destination folder according to your liking and uncheck “Add anaconda to my PATH environment variable.”

Apache Cassandra https://downloads.datastax.com/#ddacs

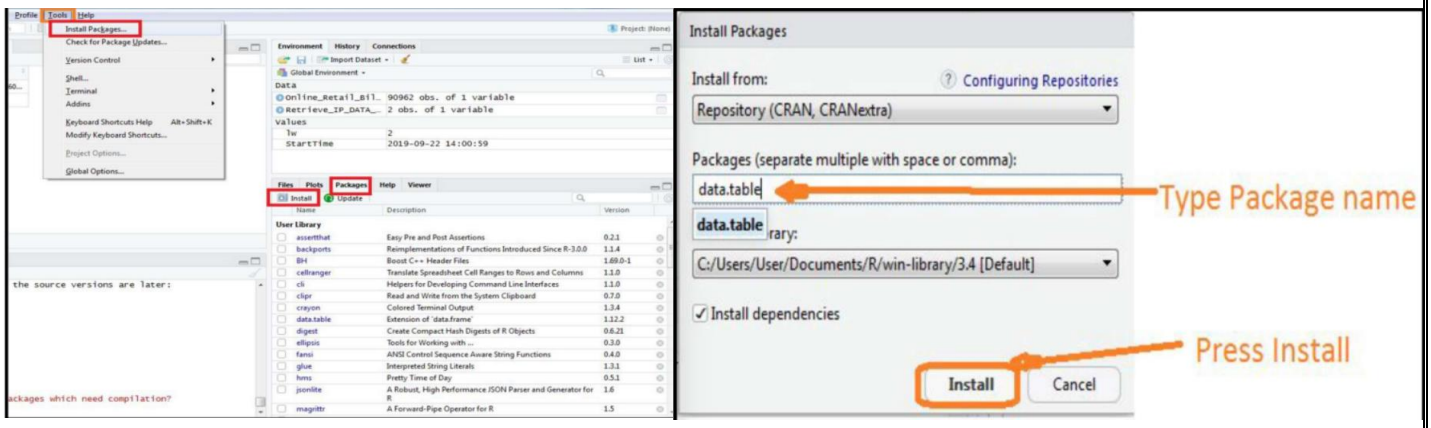
There is a dependency on the Visual C++ 2008 runtime (32bit), but Windows 7 and Windows 2008 Server R2 has it already installed. Download it from: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=29

JDK 1.8 - Sypder

If Working on Windows OS, create a Directory C:/VKHCG.

R Packages :-R- Studio

R – Studio: Go to Tools → Install Packages OR Select Install tab from Package Tab



Then type the package name in package text field

Install following package

- Data.Table Package
- ReadR Package
- JSONLite Package
- Ggplot2 Package
- Sparklyr Package
- Tibble package

R - Console

Use the following command:

- install.packages ("data.table")
- install.packages("readr")
- install.packages ("jsonlite")
- install.packages("ggplot2")
- install.packages("sparklyr")
- install.packages(“tibble”)

Practical NO: - 02

Aim: - Creating database using Cassandra

Creating and using database in Cassandra

```
cqlsh> CREATE KEYSPACE mydb WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replica
tion_factor':1}
   ... ;
cqlsh> use mydb
   ... ;
```

Create Database mydb and table books to store and retrieve records. Also update and delete records

```
cqlsh> CREATE KEYSPACE mydb WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replica
tion_factor':1}
   ... ;
cqlsh> use mydb
   ... ;
cqlsh:mydb> CREATE TABLE books (id int PRIMARY KEY, title text, year text);
cqlsh:mydb> DESC books;

CREATE TABLE mydb.books (
    id int PRIMARY KEY,
    title text,
    year text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompacti
onStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.
compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:mydb> Insert into books(id,title,year) values (1,‘Data Science’,’2020’);
cqlsh:mydb> Insert into books(id,title,year) values (2,‘Machine Learning’,’2020’);
cqlsh:mydb> Insert into books(id,title,year) values (3,‘Artifical Intelligence’,’2020
’);
cqlsh:mydb> Select * from books;

 id | title                  | year
----+------------------------+------
  1 |           Data Science | 2020
  2 |       Machine Learning | 2020
  3 | Artifical Intelligence | 2020

(3 rows)
cqlsh:mydb> delete from books where id=3;
cqlsh:mydb> Select * from books;

 id | title            | year
----+------------------+------
  1 |     Data Science | 2020
  2 | Machine Learning | 2020

(2 rows)
cqlsh:mydb> update books set year=’2021’ where id=1;
cqlsh:mydb> Select * from books;

 id | title            | year
----+------------------+------
  1 |     Data Science | 2021
  2 | Machine Learning | 2020

(2 rows)
cqlsh:mydb>
```

Create Database mydb1 and table employee to store and retrieve records.

```
cqlsh> CREATE KEYSPACE mydb1 WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' :1};
cqlsh> use mydb1;
cqlsh:mydb1> CREATE TABLE emp (emp_id int PRIMARY KEY, emp_name text, dept_id int,email text,phone text);
cqlsh:mydb1> Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1001, 'ABCD', 1001, 'abcd@company.com', '1122334455');
cqlsh:mydb1> Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1002, 'DEFG', 1001, 'defg@company.com', '2233445566');
cqlsh:mydb1> Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1003, 'GHIJ', 1002, 'ghij@company.com', '3344556677');
cqlsh:mydb1> Select *from emp
        ... ;

 emp_id | dept_id | email           | emp_name | phone
--------+---------+-----------------+----------+------------
   1001 |    1001 | abcd@company.com |     ABCD | 1122334455
   1003 |    1002 | ghij@company.com |     GHIJ | 3344556677
   1002 |    1001 | defg@company.com |     DEFG | 2233445566

(3 rows)
cqlsh:mydb1>
```

Use Database mydb1 and table student to store and retrieve records.

```
cqlsh:mydb1> CREATE TABLE student (stu_id int PRIMARY KEY, name text,course text,duration text);
cqlsh:mydb1> Insert into student ( stu_id, name, course, duration ) values (100, 'khushi', 'MSC.IT', '2 Years');
cqlsh:mydb1> Insert into student ( stu_id, name, course, duration ) values (101, 'jeni', 'MSC.DS', '2 Years');
cqlsh:mydb1> Insert into student ( stu_id, name, course, duration ) values (102, 'soni', 'BSC.DS', '3 Years');
cqlsh:mydb1> select *from student
        ... ;

 stu_id | course | duration | name
--------+--------+----------+--------
    100 | MSC.IT |  2 Years | khushi
    102 | BSC.DS |  3 Years |   soni
    101 | MSC.DS |  2 Years |   jeni

(3 rows)
```

Practical NO :- 03

Aim:- Write the programs to convert Text Delimited CSV to HORUS format.

Code:-

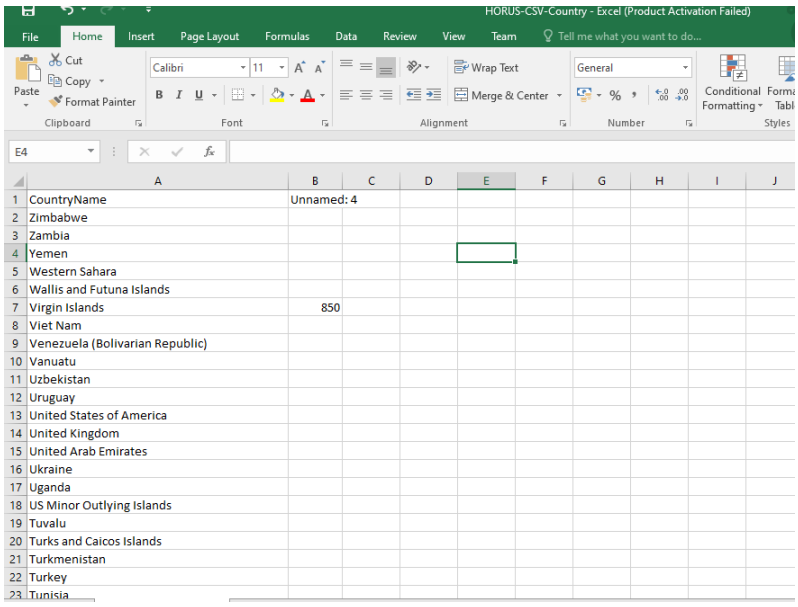
```
import pandas as pd
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
ProcessData=InputData
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
ProcessData.set_index('CountryNumber', inplace=True)
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values ================================')
print(ProcessData)
print('=====================================================')
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
```

Output:- Before

```
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
5                       Andorra                  AD        AND     20
6                        Angola                  AO        AGO     24
7                      Anguilla                  AI        AIA    660
8                    Antarctica                  AQ        ATA     10
9             Antigua and Barbuda                  AG        ATG     28
10                      Argentina                  AR        ARG     32
11                      Armenia                  AM        ARM     51
12                        Aruba                  AW        ABW    533
13                    Australia                  AU        AUS     36
14                      Austria                  AT        AUT     40
15                   Azerbaijan                  AZ        AZE     31
16                      Bahamas                  BS        BHS     44
```

After:

```
In [4]:
245         NaN
246         NaN

[247 rows x 5 columns]
====================================================
Process Data Values ================================
                                  CountryName  Unnamed: 4
CountryNumber
716                                  Zimbabwe         NaN
894                                    Zambia         NaN
887                                     Yemen         NaN
732                           Western Sahara         NaN
876             Wallis and Futuna Islands            NaN
VIR                           Virgin Islands       850.0
704                                 Viet Nam         NaN
862          Venezuela (Bolivarian Republic)         NaN
548                                  Vanuatu         NaN
860                                Uzbekistan         NaN
858                                  Uruguay         NaN
840                 United States of America         NaN
```



Practical NO :- 04

Aim:- Write the programs to convert XML to HORUS format.

Code:-

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

```
    return pd.DataFrame(all_records)
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()
print('=================================================')
print('Input Data Values ===============================')
print('=================================================')
print(InputData)
print('=================================================')
ProcessDataXML=InputData
ProcessData=xml2df(ProcessDataXML)
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
ProcessData.set_index('CountryNumber', inplace=True)
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('=================================================')
print('Process Data Values ==============================')
print('=================================================')
print(ProcessData)
print('=================================================')
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=================================================')
print('XML to HORUS - Done')
print('=================================================')
```

Output:- Before

```
==================================================
Input Data Values ================================
==================================================
<root><entry><Country>Afghanistan</Country><Country>Afghanistan</Country><ISO-2-CODE>AF</ISO-2-CODE><ISO-2-CODE>AF</ISO-2-COD
E><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>Ala
nd Islands</Country><Country>Aland Islands</Country><ISO-2-CODE>AX</ISO-2-CODE><ISO-2-CODE>AX</ISO-2-CODE><ISO-3-Code>ALA</IS
O-3-Code><ISO-3-Code>ALA</ISO-3-Code><ISO-M49>248</ISO-M49><ISO-M49>248</ISO-M49></entry><entry><Country>Albania</Country><Co
untry>Albania</Country><ISO-2-CODE>AL</ISO-2-CODE><ISO-2-CODE>AL</ISO-2-CODE><ISO-3-Code>ALB</ISO-3-Code><ISO-3-Code>ALB</ISO
-3-Code><ISO-M49>8</ISO-M49><ISO-M49>8</ISO-M49></entry><entry><Country>Algeria</Country><Country>Algeria</Country><ISO-2-COD
E>DZ</ISO-2-CODE><ISO-2-CODE>DZ</ISO-2-CODE><ISO-3-Code>DZA</ISO-3-Code><ISO-3-Code>DZA</ISO-3-Code><ISO-M49>12</ISO-M49><ISO
-M49>12</ISO-M49></entry><entry><Country>American Samoa</Country><Country>American Samoa</Country><ISO-2-CODE>AS</ISO-2-CODE>
<ISO-2-CODE>AS</ISO-2-CODE><ISO-3-Code>ASM</ISO-3-Code><ISO-3-Code>ASM</ISO-3-Code><ISO-M49>16</ISO-M49><ISO-M49>16</ISO-M49>
</entry><entry><Country>Andorra</Country><Country>Andorra</Country><ISO-2-CODE>AD</ISO-2-CODE><ISO-2-CODE>AD</ISO-2-CODE><ISO
-3-Code>AND</ISO-3-Code><ISO-3-Code>AND</ISO-3-Code><ISO-M49>20</ISO-M49><ISO-M49>20</ISO-M49></entry><entry><Country>Angola
</Country><Country>Angola</Country><ISO-2-CODE>AO</ISO-2-CODE><ISO-2-CODE>AO</ISO-2-CODE><ISO-3-Code>AGO</ISO-3-Code><ISO-3-C
ode>AGO</ISO-3-Code><ISO-M49>24</ISO-M49><ISO-M49>24</ISO-M49></entry><entry><Country>Anguilla</Country><Country>Anguilla</Co
untry><ISO-2-CODE>AI</ISO-2-CODE><ISO-2-CODE>AI</ISO-2-CODE><ISO-3-Code>AIA</ISO-3-Code><ISO-3-Code>AIA</ISO-3-Code><ISO-M49>
660</ISO-M49><ISO-M49>660</ISO-M49></entry><entry><Country>Antarctica</Country><Country>Antarctica</Country><ISO-2-CODE>AQ</I
SO-2-CODE><ISO-2-CODE>AQ</ISO-2-CODE><ISO-3-Code>ATA</ISO-3-Code><ISO-3-Code>ATA</ISO-3-Code><ISO-M49>10</ISO-M49><ISO-M49>10
```

After:-

CountryNumber	CountryName
716	Zimbabwe
894	Zambia
887	Yemen
732	Western Sahara
876	Wallis and Futuna Islands
850	Virgin Islands, US
704	Viet Nam
862	Venezuelaÿ(Bolivarian Republic)
548	Vanuatu
860	Uzbekistan
858	Uruguay
840	United States of America
826	United Kingdom
784	United Arab Emirates
804	Ukraine
800	Uganda
581	US Minor Outlying Islands

s

Practical NO :- 05

Aim:- Write the programs to convert JSON to HORUS format.

Code:-

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

Output:-

```
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
```

Practical NO :- 06

Aim:- Write the programs to convert MySQL to HORUS format.

Code:-

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

```
OutputData.to_csv(sOutputFileName, index = False)
print('Database to HORUS - Done')
```

Output:-

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

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

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

Practical NO :- 07

Aim:- Write the programs to convert Picture (JPEG) to HORUS format.

Code:-

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

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

Output:-

```
Input Data Values ==================================
X:  800
Y:  1200
RGBA:  4
=====================================================
Rows:  640000
Columns : 6
=====================================================
Process Data Values ================================
=====================================================
```


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

Practical NO :- 08

Aim:- Write the programs to convert Video to HORUS format.

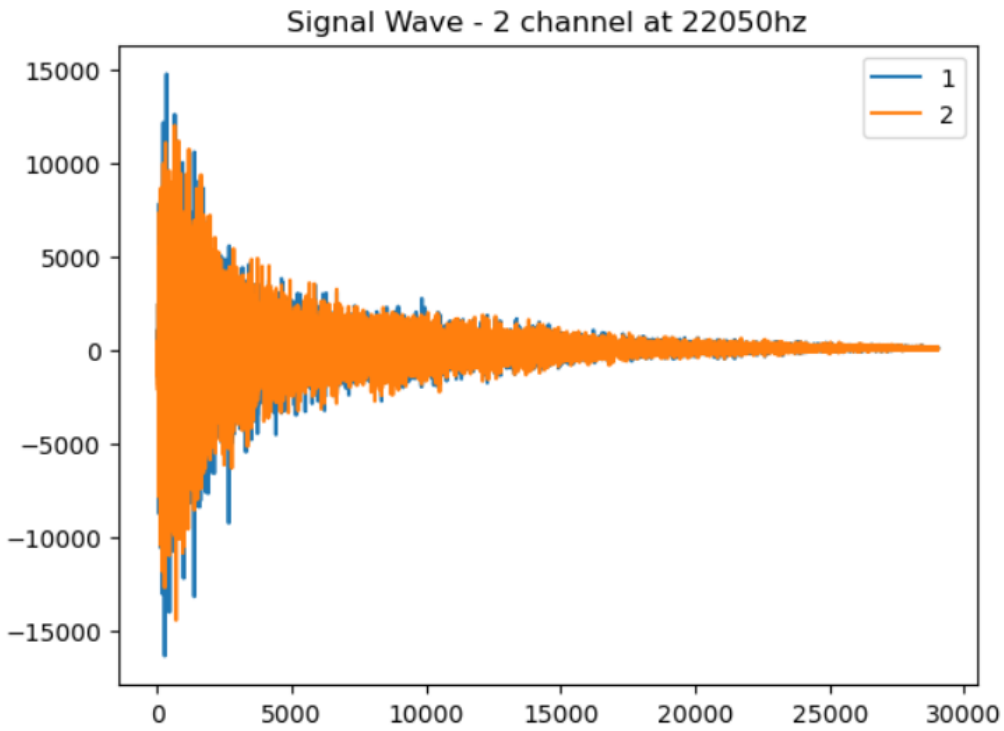
Code:-

```
from __future__ import with_statement
from PIL import Image # pip install Pillow
import cv2 # pip install opencv-python
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')
```

Output:-

```
=====================================================
Processing :  C:/VKHCG/2ch-sound.wav
=====================================================
----------------
Audio: 2 channel
----------------
Rate: 22050
----------------
shape: (29016, 2)
dtype: int16
min, max: -16384 14767
----------------
```



Practical NO :- 09

Aim:- Write the programs to convert Audio to HORUS format.

Code:-

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

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

```
sColumns = ["Ch1", "Ch2", "Ch3", "Ch4", "Ch5", "Ch6", "Ch7", "Ch8"]
ProcessData.columns = sColumns
OutputData = ProcessData
sOutputFileName = "Audio-to-HORUS-outputG-8ch.csv"
print("====================================================")
print("Audio to HORUS - Done")
OutputData.to_csv(sOutputFileName, index=False)
```

Output:-

```
====================================================
Processing :  C:/VKHCG/05-DS/9999-Data/2ch-sound.wav
====================================================
----------------
Audio: 2 channel
----------------
Rate: 22050
----------------
shape: (29016, 2)
dtype: int16
min, max: -16384 14767
----------------

<Figure size 640x480 with 1 Axes>

====================================================
Processing :  C:/VKHCG/05-DS/9999-Data/4ch-sound.wav
====================================================
----------------
Audio: 4 channel
----------------
Rate: 44100
----------------
shape: (169031, 4)
dtype: int16
min, max: -31783 26018
----------------

<Figure size 640x480 with 1 Axes>
```

Practical NO :- 10

Aim:- Write the program to use fixers utilities to solve quality issues.

Code:-

```
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print('#1 Removing leading or lagging spaces from a data entry');
baddata = " Hello My name is abc khushi kunti "
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(2001, 1, 1)
baddata=format(baddate,'%Y-%m-%d')
gooddate = dt.datetime.strptime(baddata,'%Y-%m-%d')
gooddata=format(gooddate,'%d %B %Y')
print('Bad Data : ',baddata)
```

print('Good Data : ',gooddata)

Output:-

```
4A. Fixers Utilities:
#1 Removing leading or lagging spaces from a data entry
>  Hello My name is abc khushi kunti  <
> Hello My name is abc khushi kunti <
**********************************************************
#2 Removing nonprintable characters from a data entry
Bad Data :  DataScience with funny characters is bad!!!
Clean Data :  DataScience with funny characters is bad!!!
**********************************************************
# 3 Reformatting data entry to match specific formatting criteria.
Bad Data :  2001-01-01
Good Data :  01 January 2001
```

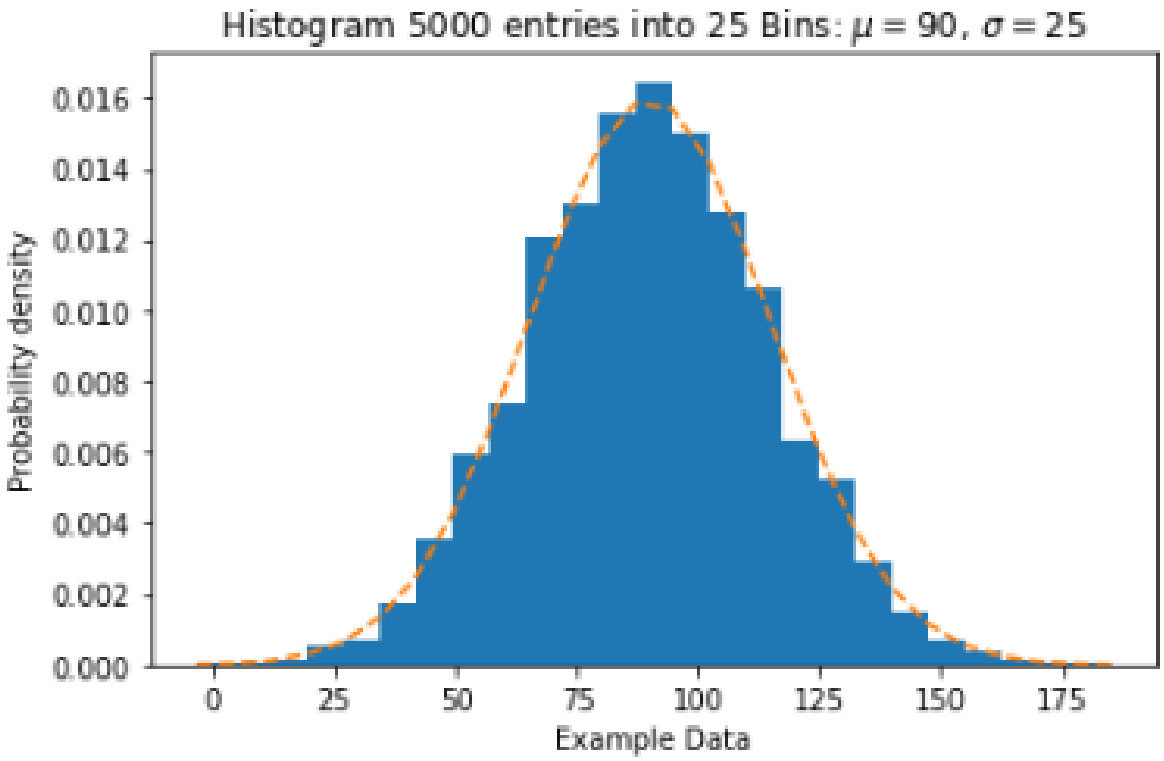
Practical NO :- 11

Aim:- Write the program to use data binning or bucketing to reduce the effects of minor observation error.

Code:-

```
import numpy as np

import matplotlib.mlab as mlab

import matplotlib.pyplot as plt

import scipy.stats as stats

np.random.seed(0)

mu = 90 # mean of distribution

sigma = 25 # standard deviation of distribution

x = mu + sigma * np.random.randn(5000)

num_bins = 25

fig, ax = plt.subplots()

# the histogram of the data

n, bins, patches = ax.hist(x, num_bins, density=1)

# add a 'best fit' line

y = stats.norm.pdf(bins, mu, sigma)

# mlab.normpdf(bins, mu, sigma)

ax.plot(bins, y, '--')

ax.set_xlabel('Example Data')

ax.set_ylabel('Probability density')

sTitle=r'Histogram ' + str(len(x)) + ' entries into ' + str(num_bins) + ' Bins: $\mu=' + str(mu) + '$, $\sigma=' + str(sigma) + '$'

ax.set_title(sTitle)

fig.tight_layout()

sPathFig='C:/VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png'

fig.savefig(sPathFig)

plt.show()
```

Output:-


Histogram 5000 entries into 25 Bins: $\mu = 90$, $\sigma = 25$
Probability density
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
25
50
75
100
125
150
175
Example Data

Practical NO :- 12

Aim:- Write the program to demonstrate averaging of data.

Code:-

```
import pandas as pd
InputFileName='IP_DATA_CORE2.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('4C. Averaging of Data')
print('Working Base :',Base, ' using ')
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
AllData=IP_DATA_ALL[['Country', 'Place_Name','Latitude']]
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
print(MeanData)
```

Output:-

```
################################
Working Base : C:/VKHCG  using
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE2.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
5         US   New York   40.7528
6         US   New York   40.7528
7         US   New York   40.7528
8         US   New York   40.7528
9         US   New York   40.7528
10        US   New York   40.7528
11        US   New York   40.7528
12        US   New York   40.7528
```

Practical NO :- 13

Aim:- Write the program to use outlier detection to find different data that may cause error.

Code:-

```
import pandas as pd
print('4D. Outlier Detection')
InputFileName='IP_DATA_CORE2.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base)
print('##################################')
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data')
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
```

```
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) & (AllData.Latitude<=UpperBound)]
print(OutliersNot)
```

Output:-

```
##################################
Working Base : C:/VKHCG
##################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE2.csv
All Data
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
1915      GB     London   51.4739
1916      GB     London   51.5491
1917      GB     London   51.5085
1918      GB     London   51.5085
1919      GB     London   51.5161
1920      GB     London   51.5198
1921      GB     London   51.5198
```

Practical NO :- 14

Aim:- Write the python program to do basic logging in data science.

Code:-

```
import sys
import os
import logging
import uuid
import shutil
import time
if sys.platform == "linux":
    Base = os.path.expanduser("~") + "/VKHCG"
else:
    Base = "C:/VKHCG"
sCompanies = ["01-Vermeulen", "02-Krennwallner", "03-Hillman", "04-Clark"]
sLayers = [
    "01-Retrieve",
    "02-Assess",
    "03-Process",
    "04-Transform",
    "05-Organise",
    "06-Report",
]
sLevels = ["debug", "info", "warning", "error"]
for sCompany in sCompanies:
    sFileDir = Base + "/" + sCompany
    if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
    for sLayer in sLayers:
        log = logging.getLogger()  # root logger
```

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

```
# Now, we can log to the root logger, or any other logger. First the root...
logging.info("Practical Data Science is fun!.")
for sLevel in sLevels:
    sApp = "Apllication-" + sCompany + "-" + sLayer + "-" + sLevel
    logger = logging.getLogger(sApp)
    if sLevel == "debug":
        logger.debug("Practical Data Science logged a debugging message.")
    if sLevel == "info":
        logger.info("Practical Data Science logged information message.")
    if sLevel == "warning":
        logger.warning("Practical Data Science logged a warning message.")
    if sLevel == "error":
        logger.error("Practical Data Science logged an error message.")
```

Output:-

```
root        : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-01-Retrieve-info: INFO     Practical Data Science logged information message.
Apllication-01-Vermeulen-01-Retrieve-warning: WARNING  Practical Data Science logged a warning message.
Apllication-01-Vermeulen-01-Retrieve-error: ERROR    Practical Data Science logged an error message.

Set up: C:/VKHCG/01-Vermeulen/01-Retrieve/Logging/Logging_c2eb8f4d-a51d-4c03-a707-1a9250e21924.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:/VKHCG/01-Vermeulen/02-Assess/Logging/Logging_a980041a-594e-4626-8857-13de83ed91e8.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:/VKHCG/01-Vermeulen/03-Process/Logging/Logging_2ba7c915-4bbb-4dbc-ba57-0405c0237fc2.log
```

Practical NO :- 15

Aim:- Write the program to retrieve different attributes of the data

Code:-

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

```
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))

#print(IP_DATA_ALL_FIX.head())

print('Fixed Data Set with ID')

IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX

IP_DATA_ALL_with_ID.index.names = ['RowID']

#print(IP_DATA_ALL_with_ID.head())

sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'

IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")

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

Output:-

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

Practical NO :- 16

Aim:- Write the program to determine the pattern of data values using R

Code:-

```
> library(readr)

> library(data.table)

data.table 1.16.0 using 6 threads (see ?getDTthreads).  Latest news: r-datatable.com

> FileName=paste0('c:/VKHCG/IP_DATA_ALL.csv')

> IP_DATA_ALL <- read_csv(FileName)
[1mindexing[0m [34mIP_DATA_ALL.csv[0m [----------------------------------] [32m459.92MB/s[0m, eta:
[36m 0s[0m
[1mindexing[0m [34mIP_DATA_ALL.csv[0m [================================---]
[32m442.84MB/s[0m, eta: [36m 0s[0m
[1mindexing[0m [34mIP_DATA_ALL.csv[0m [=================================-]
[32m452.20MB/s[0m, eta: [36m 0s[0m
[1mindexing[0m [34mIP_DATA_ALL.csv[0m [==================================]
[32m449.49MB/s[0m, eta: [36m 0s[0m
New names:

• `` -> `...1`

Rows: 1247502 Columns: 9

── Column specification
--------------------------------------------------------------------------------

Delimiter: ","

chr (3): Country, Place.Name, Post.Code

dbl (6): ...1, ID, Latitude, Longitude, First.IP.Number, Last.IP.Number

ℹ Use `spec()` to retrieve the full column specification for this data.

ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.

> hist_country=data.table(Country=unique(IP_DATA_ALL$Country))

> pattern_country=data.table(Country=hist_country$Country,

+                            PatternCountry=hist_country$Country)

> oldchar=c(letters,LETTERS)

> newchar=replicate(length(oldchar),"A")
```

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

Output:-

Data: 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
17	SZ	AA
18	MG	AA
19	ML	AA
20	TN	AA
21	MU	AA
22	MW	AA

Practical NO:- 17

Aim:- Write the program to load data set containing different ip addresses allocation.

Code:-

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

Output:-

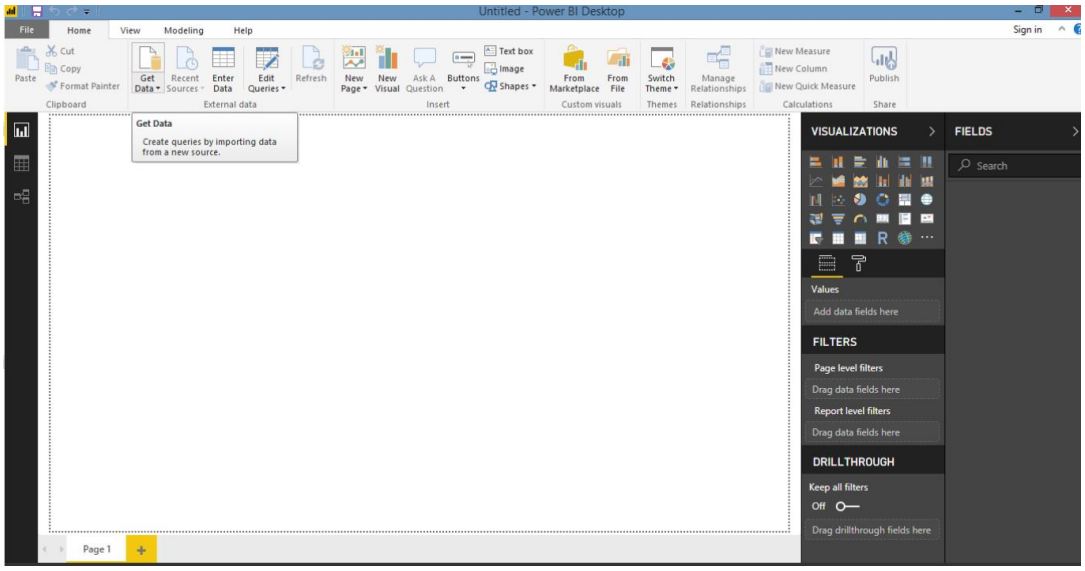
```
Loading : C:/VKHCG/IP_DATA_ALL.csv
Rows: 1247502
Columns: 9
### Raw Data Set #####################################
Unnamed: 0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class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

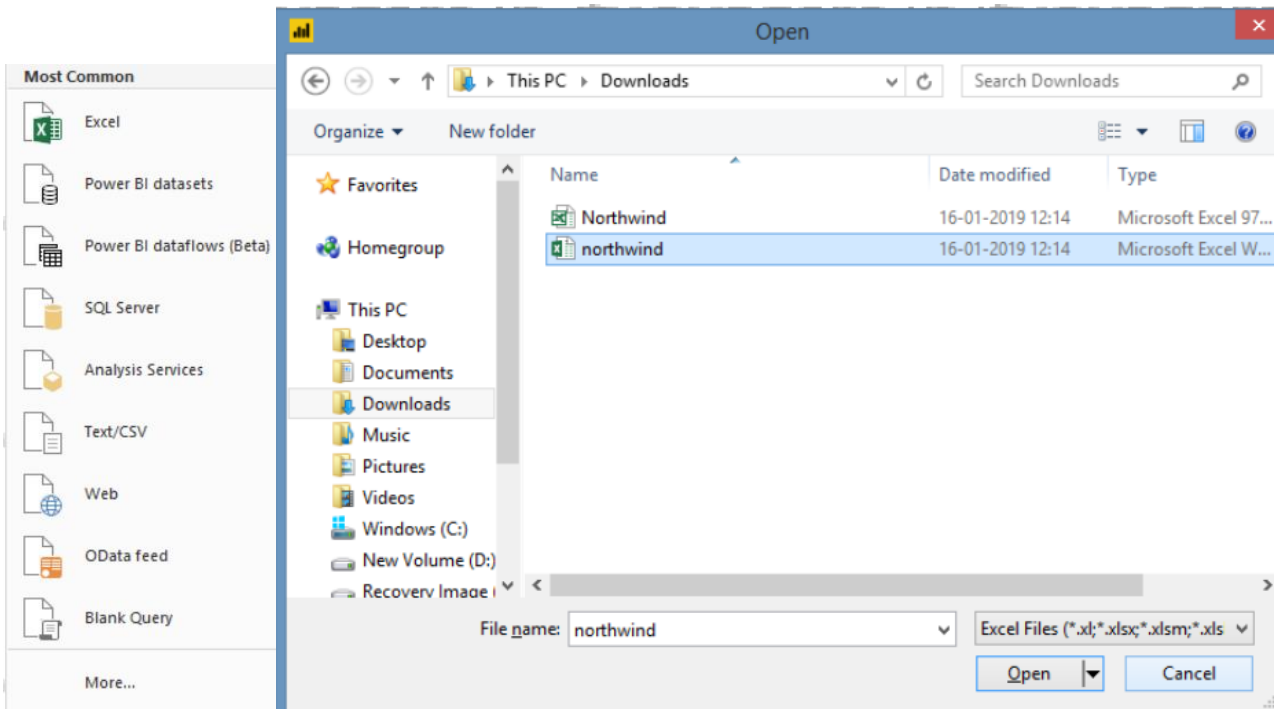
Practical NO:- 18

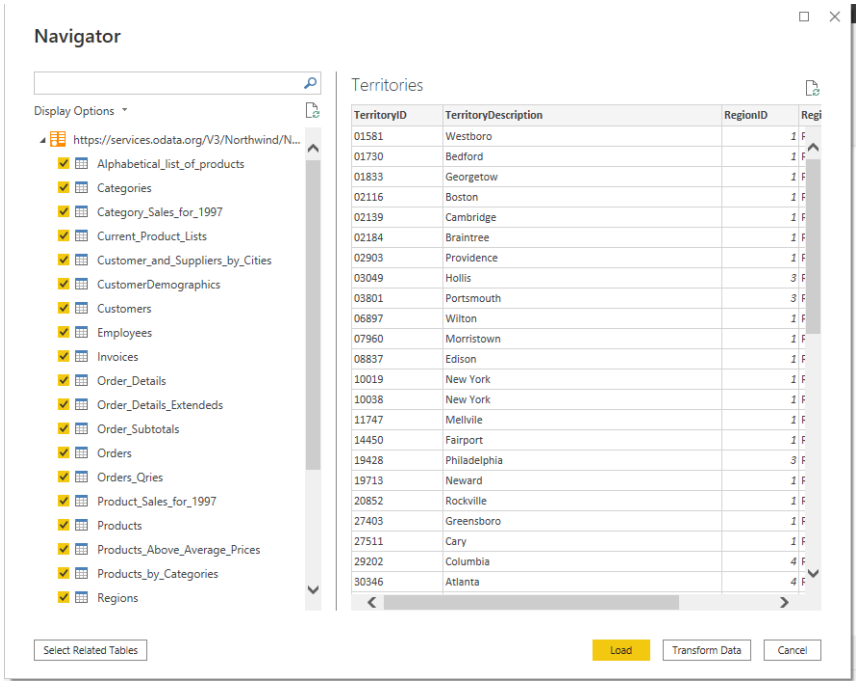
Aim:- Demonstrate organizing data using power BI.

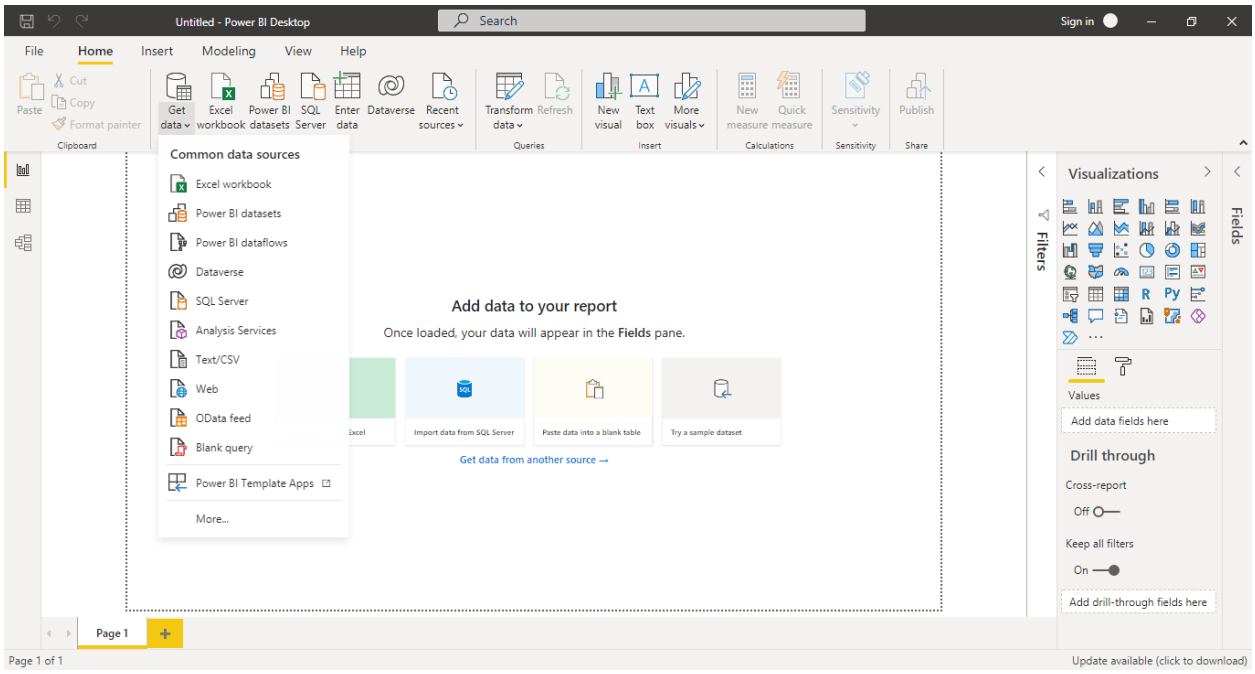
To effectively organize and analyze data using Power BI, follow these structured steps:

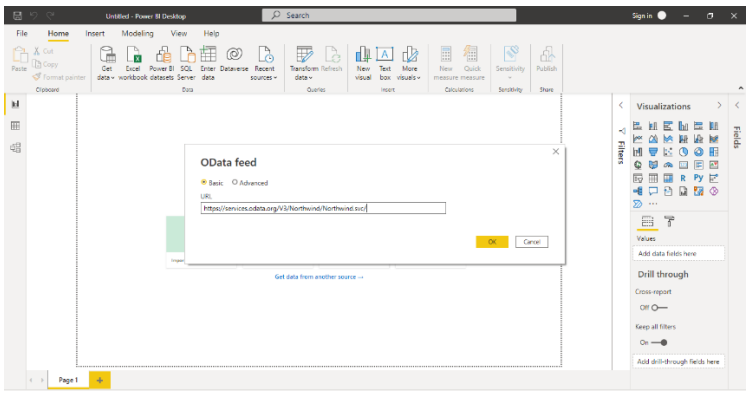
Data Acquisition: Import data from sources like Excel, CSV, or databases using the 'Get Data' feature.

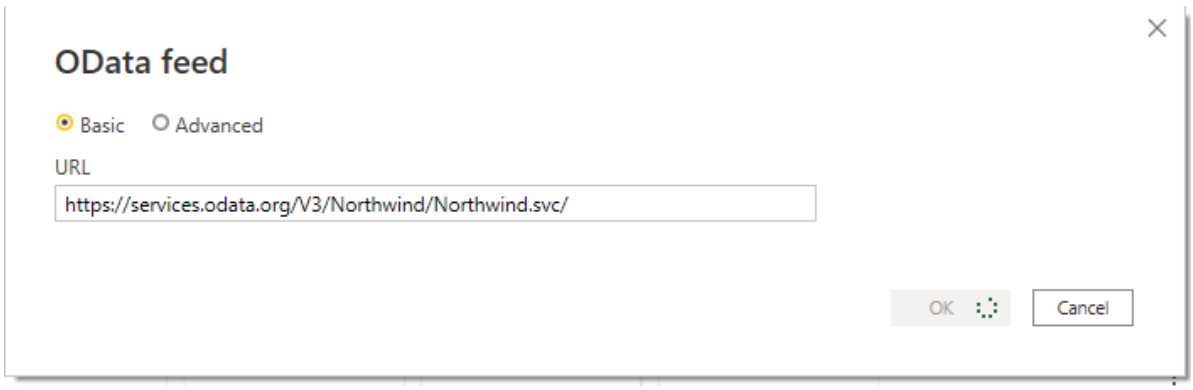


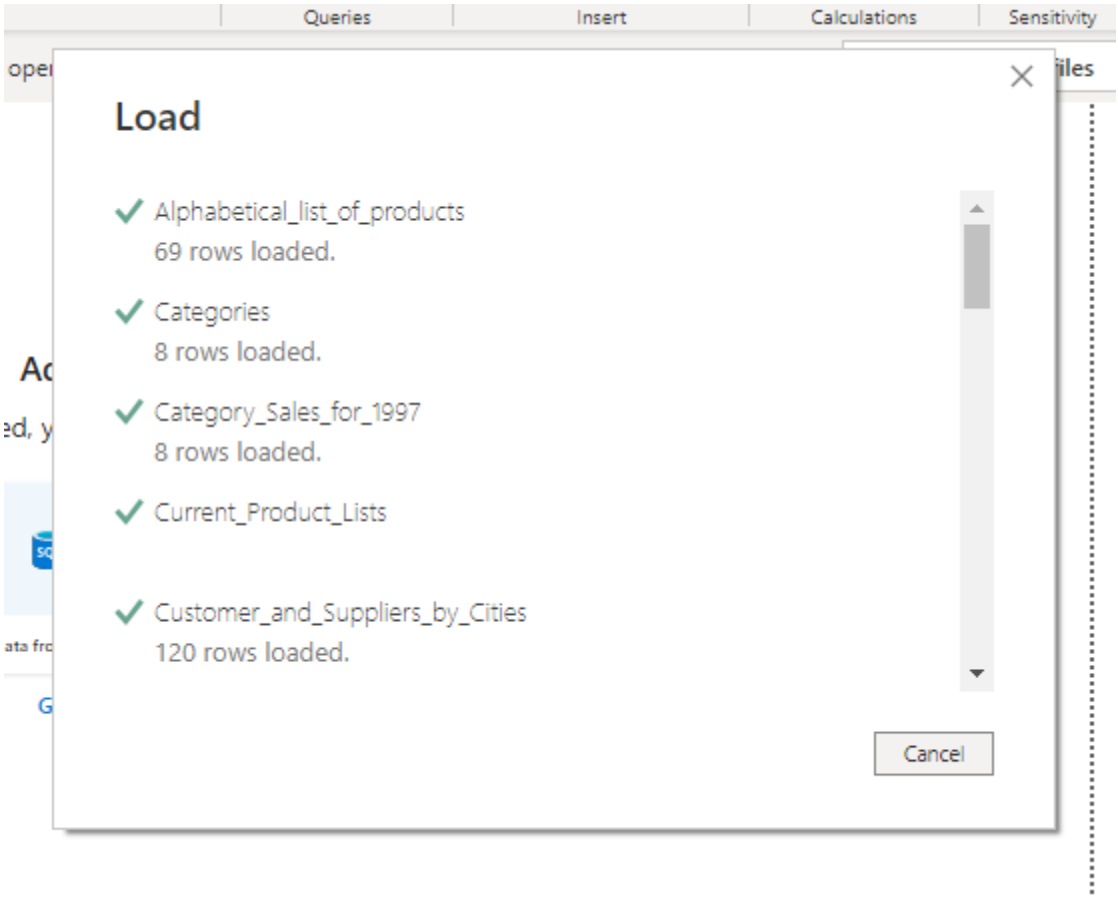



Navigator
Display Options
https://services.odata.org/V3/Northwind/N...
Alphabetical_list_of_products
Categories
Category_Sales_for_1997
Current_Product_Lists
Customer_and_Suppliers_by_Cities
CustomerDemographics
Customers
Employees
Invoices
Order_Details
Order_Details_Extendeds
Order_Subtotals
Orders
Orders_Qries
Product_Sales_for_1997
Products
Products_Above_Average_Prices
Products_by_Categories
Regions
Territories
TerritoryID TerritoryDescription RegionID
01581 Westboro 1
01730 Bedford 1
01833 Georgetow 1
02116 Boston 1
02139 Cambridge 1
02184 Braintree 1
02903 Providence 1
03049 Hollis 3
03801 Portsmouth 3
06897 Wilton 1
07960 Morristown 1
08837 Edison 1
10019 New York 1
10038 New York 1
11747 Mellvile 1
14450 Fairport 1
19428 Philadelphia 3
19713 Neward 1
20852 Rockville 1
27403 Greensboro 1
27511 Cary 1
29202 Columbia 4
30346 Atlanta 4
Select Related Tables
Load
Transform Data
Cancel


Untitled - Power BI Desktop
Search
Sign in
File Home Insert Modeling View Help
Paste Cut Copy Format painter
Get data Excel workbook Power BI datasets SQL Server Enter data Dataverse Recent sources
Transform data Refresh
New visual Text box More visuals
New measure Quick measure
Sensitivity
Publish
Clipboard Queries Insert Calculations Sensitivity Share
Common data sources
Excel workbook
Power BI datasets
Power BI dataflows
Dataverse
SQL Server
Analysis Services
Text/CSV
Web
OData feed
Blank query
Power BI Template Apps
More...
Add data to your report
Once loaded, your data will appear in the Fields pane.
Import data from SQL Server
Paste data into a blank table
Try a sample dataset
Get data from another source →
Visualizations
Values
Add data fields here
Drill through
Cross-report
Off
Keep all filters
On
Add drill-through fields here
Filters
Fields
Page 1
Page 1 of 1
Update available (click to download)

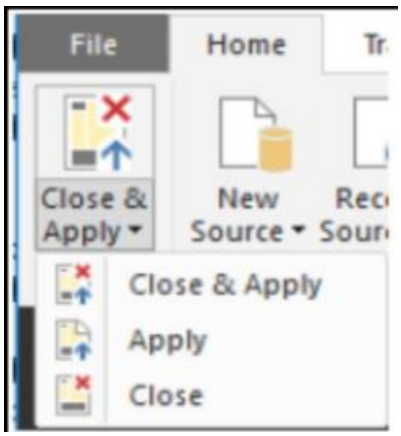

OData feed
Basic
Advanced
URL
https://services.odata.org/V3/Northwind/Northwind.svc/
OK
Cancel
Visualizations
Values
Drill through


OData feed
Basic
Advanced
URL
https://services.odata.org/V3/Northwind/Northwind.svc/
OK
Cancel

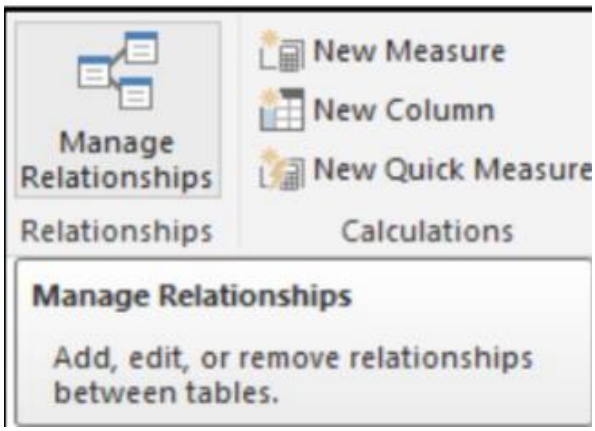

Queries
Insert
Calculations
Sensitivity
Load
Alphabetical_list_of_products
69 rows loaded.
Categories
8 rows loaded.
Category_Sales_for_1997
8 rows loaded.
Current_Product_Lists
Customer_and_Suppliers_by_Cities
120 rows loaded.
Cancel

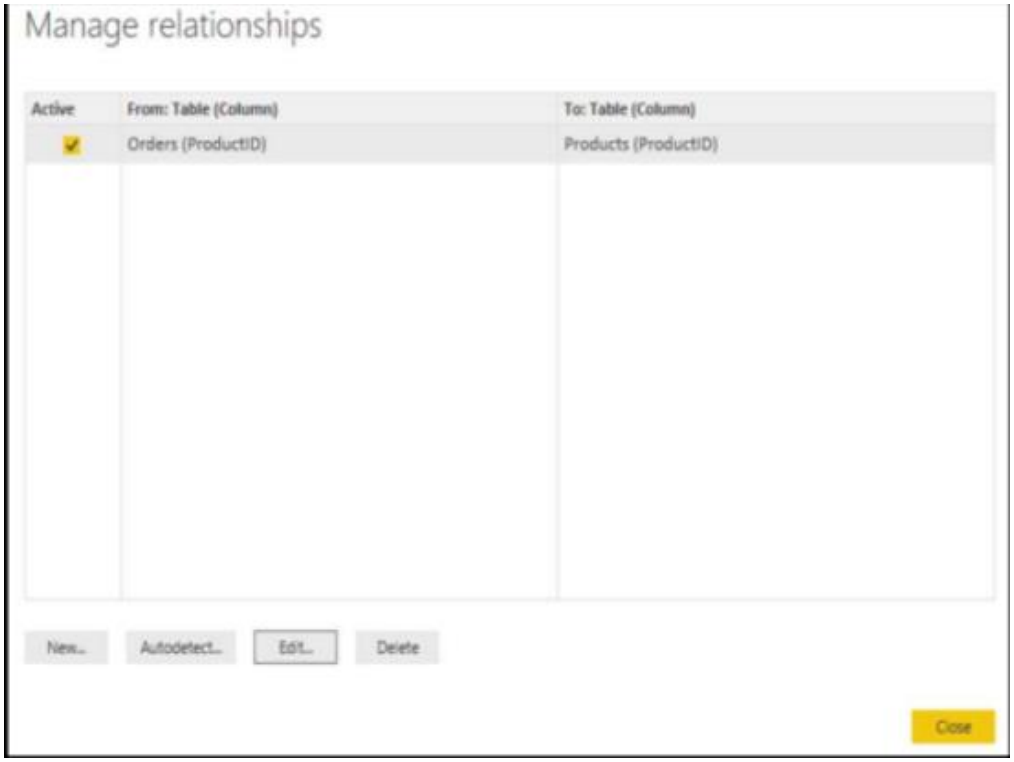
Data Cleaning and Transformation: Use the Power Query Editor to clean data — remove duplicates, handle missing values, and rename columns.

	OrderDate	ShipCity	ShipCountry	LineTotal	ProductID	UnitPrice	Quantity
1	7/4/1996 12:00:00 AM	Reims	France	168	11	14	12
2	7/4/1996 12:00:00 AM	Reims	France	98	42	9.8	10
3	7/4/1996 12:00:00 AM	Reims	France	174	72	34.8	5
4	7/5/1996 12:00:00 AM	Münster	Germany	167.4	14	18.6	9
5	7/5/1996 12:00:00 AM	Münster	Germany	1696	51	42.4	40
6	7/8/1996 12:00:00 AM	Rio de Janeiro	Brazil	77	41	7.7	10
7	7/8/1996 12:00:00 AM	Rio de Janeiro	Brazil	1484	51	42.4	35
8	7/8/1996 12:00:00 AM	Rio de Janeiro	Brazil	252	65	16.8	15
9	7/8/1996 12:00:00 AM	Lyon	France	100.8	22	16.8	6
10	7/8/1996 12:00:00 AM	Lyon	France	234	57	15.6	15
11	7/8/1996 12:00:00 AM	Lyon	France	336	65	16.8	20



Creating a Data Model: Establish relationships between tables based on keys and choose an appropriate model type (e.g., Star Schema).





Edit relationship

Select tables and columns that are related.

Orders

OrderDate	ShipCity	ShipCountry	LineTotal	ProductID	UnitPrice	Quantity
10/8/1996 12:00:00 AM	Boise	USA	$291.9	16	13.9	21
10/8/1996 12:00:00 AM	Boise	USA	$1,008	35	14.4	70
10/8/1996 12:00:00 AM	Boise	USA	$288	46	9.6	30

Products

ProductID	ProductName	QuantityPerUnit	UnitsInStock
1	Chai	10 boxes x 20 bags	39
2	Chang	24 - 12 oz bottles	17
3	Aniseed Syrup	12 - 550 ml bottles	13

Cardinality: Many to one (*:1)

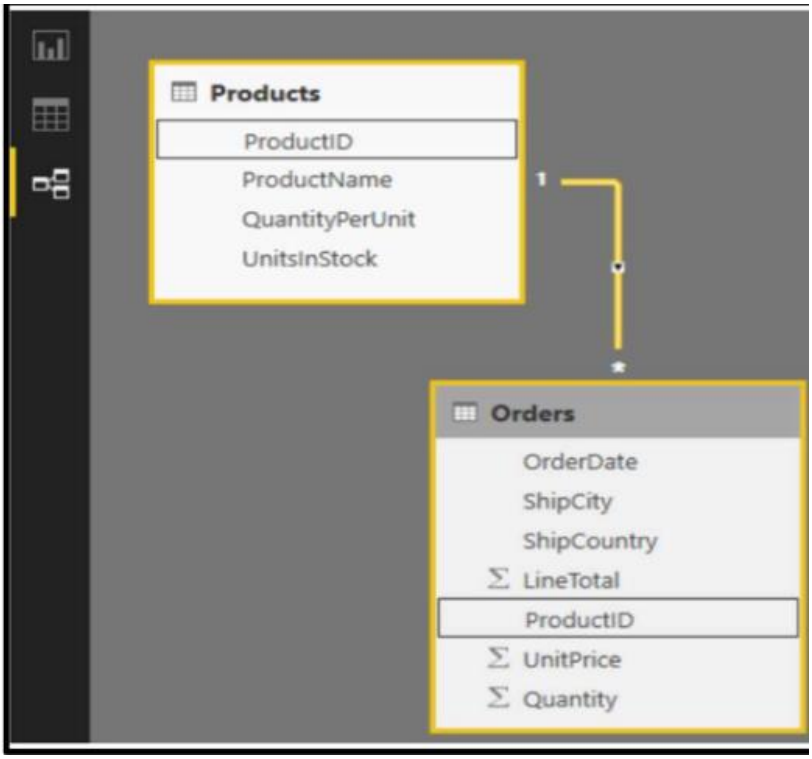
Cross filter direction: Single

☑ Make this relationship active

☐ Assume referential integrity

☐ Apply security filter in both directions

OK Cancel



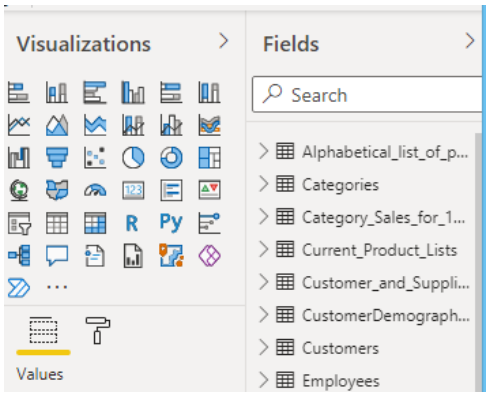
Organizing Measures: Create DAX measures for calculations and organize them into dedicated measures tables.

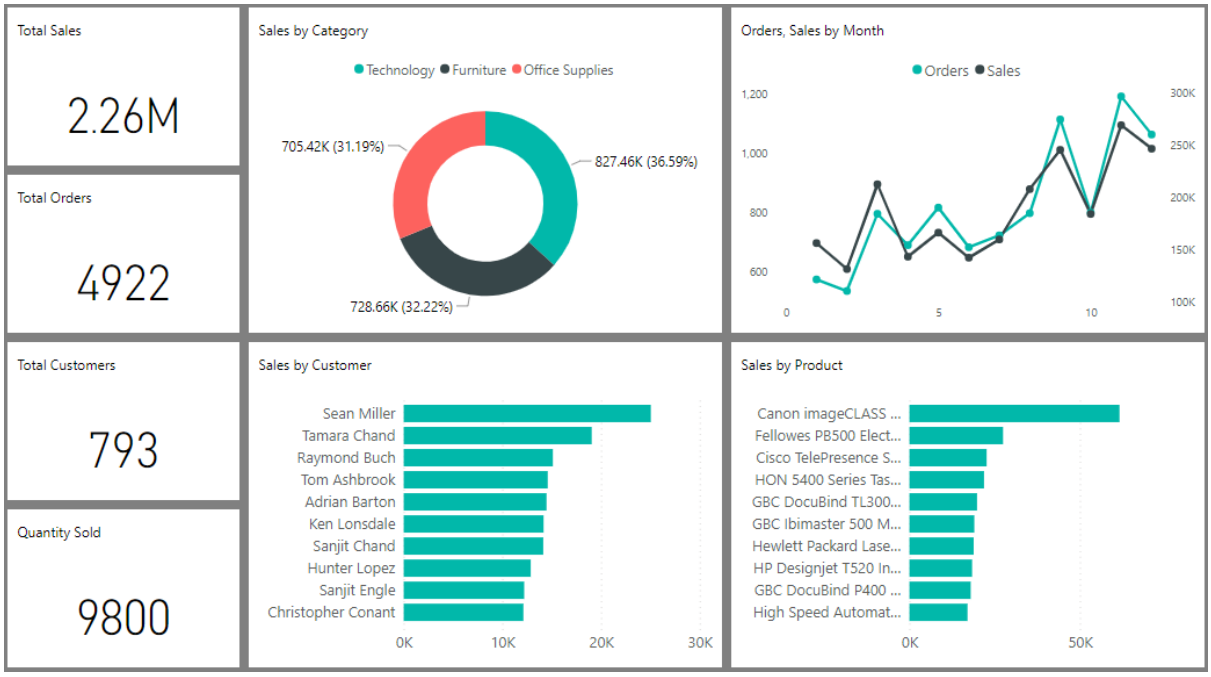
Practical NO :- 19

Aim:- Demonstrate generating data using power BI.

Setting up Filter and Fact Tables: Arrange lookup (filter) tables at the top and fact tables below for easier navigation.

Building Visualizations: Create charts, tables, and maps by dragging fields onto the report canvas.





Creating Dashboards: Pin visuals to dashboards for a consolidated view of key metrics.

Reviewing and Sharing Insights: Regularly check model accuracy and share reports/dashboards with stakeholders.

Practical NO :- 20

Aim:- Write the program to perform error management on the given data using pandas package.

Code:-

```
import sys

import os

import pandas as pd

InputFileName='DE_Billboard_Locations.csv'

OutputFileName='Retrieve_DE_Billboard_Locations.csv'

Company='02-Krennwallner'

if sys.platform == 'linux':

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

else:

    Base='C:/VKHCG'

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

Base='C:/VKHCG'

sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName

print('Loading :',sFileName)

IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,usecols=['Country','PlaceName'
,'Latitude','Longitude'])

IP_DATA_ALL.rename(columns={'PlaceName': 'Place_Name'}, inplace=True)

sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'

if not os.path.exists(sFileDir):

    os.makedirs(sFileDir)

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

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

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

sFileName2=sFileDir + '/' + OutputFileName

ROUTERLOC.to_csv(sFileName2, index = False)
```

Output:-

```
== RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python38/5d-dsprac.p
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/02-Krennwallner/00-RawData/DE_Billboard_Locations.csv
Rows : 8873
Columns : 4
### Done!! ############################################
```

Country	Place_Nar	Latitude	Longitude
DE	Lake	51.7833	8.5667
DE	Horb	48.4333	8.6833
DE	Hardenbe	51.1	7.7333
DE	Horn-bad	51.9833	8.9667
DE	Winkel	51.55	13.3833
DE	Rohrdorf	47.7333	10.0833
DE	Sch<f6>ne	51.9833	12.8333
DE	Beuren	47.75	10.0167
DE	Marienfel	50.8833	7.4333
DE	Borgholz	51.6167	9.2667
DE	Kappel	47.7667	9.45
DE	Riepe	52.9333	9.7333
DE	Lauenf<f6	51.6667	10.3833
DE	Fredeburg	51.2	8.3
DE	Dwergte	52.8833	7.9

Practical NO :- 21

Aim:- Write python/R program to pick the content for Billboard's from the given Data XML Processing.

Code:-

```
import sys
import os
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):
    try:
        root = ET.XML(xml_data)
```

```
    except ET.ParseError as e:
        print(f"XML parsing error: {e}")
        return pd.DataFrame()  # Return an empty DataFrame or handle as needed
    all_records = []
    for i, child in enumerate(root):
        record = {}
        for subchild in child:
            record[subchild.tag] = subchild.text if subchild.text else 'n/a'
        all_records.append(record)
    return pd.DataFrame(all_records)
InputFileName='IP_DATA_ALL.csv'
OutputFileName='Retrieve_Online_Visitor.xml'
CompanyIn= '01-Vermeulen'
CompanyOut= '02-Krennwallner'
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
sFileName=Base + '/' + CompanyIn + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False)
IP_DATA_ALL.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
IP_DATA_ALL.rename(columns={'First.IP.Number': 'First_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Last.IP.Number': 'Last_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Post.Code': 'Post_Code'}, inplace=True)
```

```
sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
visitordata = IP_DATA_ALL.head(10000)
print('Original Subset Data Frame')
print('Rows :',visitordata.shape[0])
print('Columns :',visitordata.shape[1])
print(visitordata)
print('Export XML')
sXML=df2xml(visitordata)
sFileName=sFileDir + '/' + OutputFileName
file_out = open(sFileName, 'wb')
file_out.write(sXML)
file_out.close()
print('Store XML:',sFileName)
xml_data = open(sFileName).read()
unxmlrawdata=xml2df(xml_data)
print('Raw XML Data Frame')
print('Rows :',unxmlrawdata.shape[0])
print('Columns :',unxmlrawdata.shape[1])
print(unxmlrawdata)
unxmldata = unxmlrawdata.drop_duplicates(subset=None, keep='first', inplace=False)
print('Deduplicated XML Data Frame')
print('Rows :',unxmldata.shape[0])
print('Columns :',unxmldata.shape[1])
print(unxmldata)
```

Output:

```
################################
Working Base : C:/VKHCG using win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Original Subset Data Frame
Rows : 10000
Columns : 8
      ID Country Place_Name ... Longitude First_IP_Number Last_IP_Number
0      1    BW  Gaborone ...  25.9119     692781056     692781567
1      2    BW  Gaborone ...  25.9119     692781824     692783103
2      3    BW  Gaborone ...  25.9119     692909056     692909311
3      4    BW  Gaborone ...  25.9119     692909568     692910079
4      5    BW  Gaborone ...  25.9119     693051392     693052415
...   ...   ...     ... ...      ...          ...           ...
9995  9996    US Waterford ... -83.3554    1144498560    1144498687
9996  9997    US Waterford ... -83.3554    1144498816    1144499199
9997  9998    US Waterford ... -83.3554    1144555136    1144555263
9998  9999    US Waterford ... -83.3554    1144881664    1144881791
9999 10000    US Waterford ... -83.3554     1171565568    1171566079

[10000 rows x 8 columns]
Export XML
Store XML: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.xml
Raw XML Data Frame
Rows : 10000
Columns : 8
      ID Country Place_Name ... Longitude First_IP_Number Last_IP_Number
0      1    BW  Gaborone ...  25.9119     692781056     692781567
1      2    BW  Gaborone ...  25.9119     692781824     692783103
2      3    BW  Gaborone ...  25.9119     692909056     692909311
3      4    BW  Gaborone ...  25.9119     692909568     692910079
4      5    BW  Gaborone ...  25.9119     693051392     693052415
...   ...   ...     ... ...      ...          ...           ...
9995  9996    US Waterford ... -83.3554    1144498560    1144498687
9996  9997    US Waterford ... -83.3554    1144498816    1144499199
9997  9998    US Waterford ... -83.3554    1144555136    1144555263
9998  9999    US Waterford ... -83.3554    1144881664    1144881791
9999 10000    US Waterford ... -83.3554     1171565568    1171566079

[10000 rows x 8 columns]
Deduplicated XML Data Frame
Rows : 10000
Columns : 8
      ID Country Place_Name ... Longitude First_IP_Number Last_IP_Number
0      1    BW  Gaborone ...  25.9119     692781056     692781567
1      2    BW  Gaborone ...  25.9119     692781824     692783103
2      3    BW  Gaborone ...  25.9119     692909056     692909311
3      4    BW  Gaborone ...  25.9119     692909568     692910079
4      5    BW  Gaborone ...  25.9119     693051392     693052415
...   ...   ...     ... ...      ...          ...           ...
9995  9996    US Waterford ... -83.3554    1144498560    1144498687
9996  9997    US Waterford ... -83.3554    1144498816    1144499199
9997  9998    US Waterford ... -83.3554    1144555136    1144555263
9998  9999    US Waterford ... -83.3554    1144881664    1144881791
9999 10000    US Waterford ... -83.3554     1171565568    1171566079

[10000 rows x 8 columns]
```

Practical NO :- 22

Aim:- Write the python/R program to create the network routing diagram from the Data on routers.

Code:-

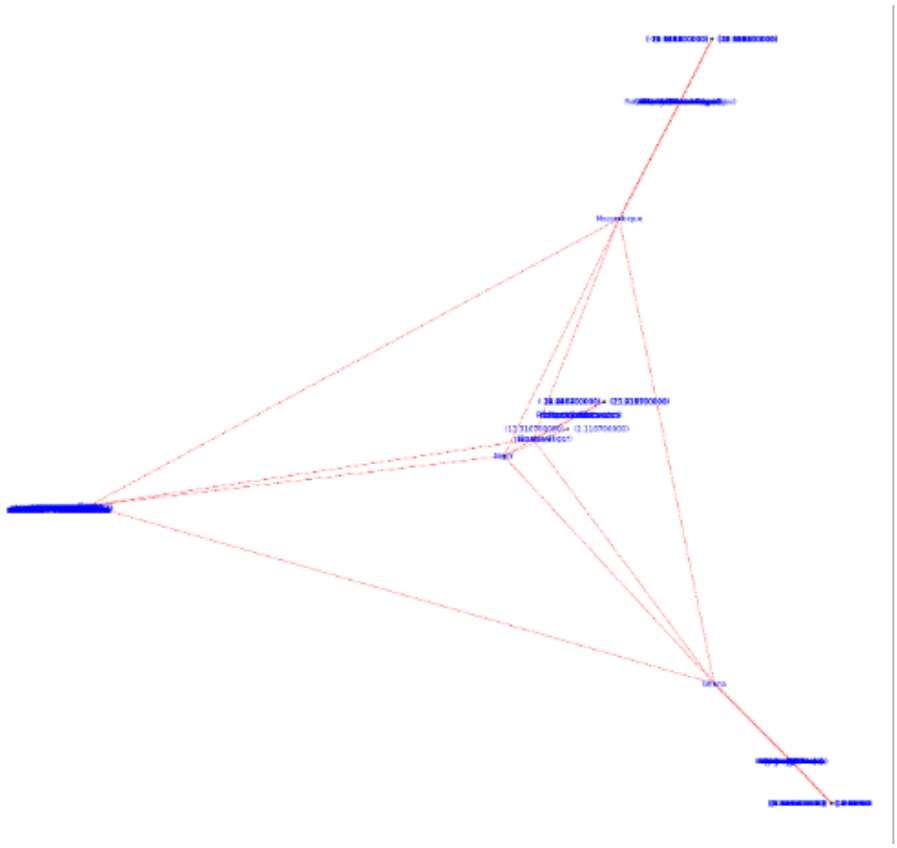
```
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
pd.options.mode.chained_assignment = None
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv'
sOutputFileName1='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml'
sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png'
Company='01-Vermeulen'
### Import Country Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
```

```
print('#################################')
CustomerDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
CustomerData=CustomerDataRaw.head(100)
print('Loaded Country:',CustomerData.columns.values)
print('#################################')
print(CustomerData.head())
print(CustomerData.shape)
G=nx.Graph()
for i in range(CustomerData.shape[0]):
    for j in range(CustomerData.shape[0]):
        Node0=CustomerData['Customer_Country_Name'][i]
        Node1=CustomerData['Customer_Country_Name'][j]
        if Node0 != Node1:
            G.add_edge(Node0,Node1)
for i in range(CustomerData.shape[0]):
    Node0=CustomerData['Customer_Country_Name'][i]
    Node1=CustomerData['Customer_Place_Name'][i] + '('+
CustomerData['Customer_Country_Name'][i] + ')'
    Node2='('+ "{:.9f}".format(CustomerData['Customer_Latitude'][i]) + ')\
    ('+ "{:.9f}".format(CustomerData['Customer_Longitude'][i]) + ')'
    if Node0 != Node1:
        G.add_edge(Node0,Node1)
    if Node1 != Node2:
        G.add_edge(Node1,Node2)
print('Nodes:', G.number_of_nodes())
```

```
print('Edges:', G.number_of_edges())
sFileName=Base + '/' + Company + '/' + sOutputFileName1
print('################################')
print('Storing :',sFileName)
print('################################')
nx.write_gml(G, sFileName)
sFileName=Base + '/' + Company + '/' + sOutputFileName2
print('################################')
print('Storing Graph Image:',sFileName)
print('################################')
plt.figure(figsize=(25, 25))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
print('################################')
print('### Done!! ####################')
print('################################')
```

Output:-

```
################################
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_Place_Name  Customer_Latitude  \
0                    BW            Gaborone           -24.6464
1                    BW         Francistown           -21.1667
2                    BW                Maun           -19.9833
3                    BW          Molepolole           -24.4167
4                    NE              Niamey            13.5167

   Customer_Longitude Customer_Country_Name
0             25.9119               Botswana
1             27.5167               Botswana
2             23.4167               Botswana
3             25.5333               Botswana
4              2.1167                  Niger
(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
################################
```



Practical NO :- 23

Aim:- Perform Linear Regression.

Code:-

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from collections import OrderedDict
person_data = []
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
        else:
            BMI_Result = 4
        person_data.append({
            'PersonID': str(t),
            'Height': height,
            'Weight': weight,
            'bmi': bmi,
            'Indicator': BMI_Result
        })
        t += 1
        print('Row:', t, 'of', tMax)
        DimPerson = pd.DataFrame(person_data)
        DimPersonIndex = DimPerson.set_index(['PersonID'], inplace=False)
    sTable = 'Transform-BMI'
    print('Storing :',sDatabaseName,'\n Table:',sTable)
    DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
    sTable = 'Person-Satellite-BMI'
    print('Storing :',sDatabaseName,'\n Table:',sTable)
    DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
    sTable = 'Dim-BMI'
    print('Storing :',sDatabaseName,'\n Table:',sTable)
    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()
diabetes = datasets.load_diabetes()
diabetes_X = diabetes.data[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-50:]
diabetes_y_train = diabetes.target[:-30]
```

```
diabetes_y_test = diabetes.target[-50:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print('Coefficients: \n', regr.coef_)
print("Mean squared error: %.2f"
% mean_squared_error(diabetes_y_test, diabetes_y_pred))
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.axis('tight')
plt.title("Diabetes")
plt.xlabel("BMI")
plt.ylabel("Age")
plt.show()
```

Output:-

```
Row: 1070 of 1080.0
Row: 1071 of 1080.0
Row: 1072 of 1080.0
Row: 1073 of 1080.0
Row: 1074 of 1080.0
Row: 1075 of 1080.0
Row: 1076 of 1080.0
Row: 1077 of 1080.0
Row: 1078 of 1080.0
Row: 1079 of 1080.0
Row: 1080 of 1080.0

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


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

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

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

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