PRACTICAL REPORT

On

Data Science

Soft Computing Techniques

Submitted in fulfilment of the

Requirement for the Award of the Degree of

M.Sc. (Information Technology) - SEM I

By

NAME : AISHWARYA MILIND DESHPANDE

ROLL NO : PSI123002

Department Of Information Technology

THE S.I.A. COLLEGE OF HIGHER EDUCATION

(Affiliated to University of Mumbai)

DOMBIVLI

MAHARASHTRA – 421203

2023-2024

THE S.I.A COLLEGE OF HIGHER EDUCATION

Affiliated to University of Mumbai

Accredited 'B+' Grade

Dombivli(E), Mumbai – 421203

Department of Information Technology

CERTIFICATE

Reaccertified that the experimental work as entered in this journal is as per syllabus in M.Sc. Information Technology for_____as prescribed by University of Mumbai and was done in the Information Technology laboratory of The S.I.A College of Higher Education by the student Mr/Ms_____having Seat No._____of class M.Sc. Information Technology - PART I during the academic year 20 -20.

No. of Experiments completed _____ out of _____

Course Coordinator
Date:

Sign of Incharge **College seal** **Sign of Examiner**

Date: **Date**

PRACTICAL REPORT

On

Data Science

Submitted in fulfilment of the

Requirement for the Award of the Degree of

M.Sc. (Information Technology) - SEM I

By

NAME : AISHWARYA MILIND DESHPANDE

ROLL NO : PSI123002

Department Of Information Technology

THE S.I.A. COLLEGE OF HIGHER EDUCATION

(Affiliated to University of Mumbai)

DOMBIVLI

MAHARASHTRA – 421203

2023– 2024

THE S.I.A COLLEGE OF HIGHER EDUCATION

Affiliated to University of Mumbai

Accredited 'B+' Grade

Dombivli(E), Mumbai – 421203

Department of Information Technology

CERTIFICATE

Reaccertified that the experimental work as entered in this journal is as per syllabus in M.Sc. Information Technology for_____as prescribed by University of Mumbai and was done in the Information Technology laboratory of The S.I.A College of Higher Education by the student Mr/Ms_____having Seat No._____of class M.Sc. Information Technology - PART I during the academic year 20 -20.

No. of Experiments completed _____ out of _____

Course Coordinator

Date:

Sign of Incharge **College seal** **Sign of Examiner**

Date: **Date**

INDEX

Sr.No	Practical	Sign
1	Creating Data Model using Cassandra.	
2	Conversion from different formats to HOURS format. **A.** Text delimited csv format. **B.** XML. **C.** JSON. **D.** MySQL Database. **E.** Video. **F.** Picture (JPEG) to HORUS format **G.** Audio.	
3	Utilities and Auditing. **A.** Fixers Utilities. **B.** Data Binning or Bucketing. **C.** Logging. **D.** Averaging of data. **E.** Outlier Detection	
4	Retrieving Data. **A.** Loading IP DATA ALL. **B.** Program to retrieve different attributes of data. **C.** Mysql. **D.** Excel.	
5	Assessing Data. **A.** Missing Values in Pandas. **1.** Drop the Columns Where All Elements Are Missing Values. **2.** Keep Only the Rows That Contain a Maximum of Two Missing Values. **3.** Fill All Missing Values with the Mean, Median, Mode, Minimum, and Maximum of the Particular Numeric Column. **4.** Write a Python/R program to build directed acyclic graph. **B.** Write python/R program to create the network routing diagram from the given data on routers **C.** Write a python/R program to build acyclic graph	

	D. Write python/R program to pick the content for BillBoards from the given data **E.** Write a python/R program to generate GML file from given csv file **F.** Write python/R program to plan location of warehouse from the given data **G.** Write python/R program using data science via clustering to determine new warehouse using the given data **H.** Using the given data Write python/R program to plan the shipping routers from best-fit international logistics **I.** Write python/R program to delete the best packing option to ship in container from the given data **J.** Write python program to create delivery route using the given data **K.** Write python program to crate simple forex trading planner from the given data **L.** Write python program to process the balance sheet to ensure the only good data is processing **M.** Write python program to generate payroll from the given data	
6	Build the time hub, links and satellites	
7	Transforming data	
8	Organizing data	
9	Generating data	
10	Data visualisation using power Bi	

Practical:1

Creating Data Model using Cassandra.

Command Prompt - cqlsh

```
(0 rows)
cqlsh:test> create table dept(dept_id int PRIMARY KEY,dept_nmae text,dept_loc text);
cqlsh:test> insert into dept(dept_id,dept_name,dept_loc) values(1001,'Accounts','Mumbai');
InvalidRequest: Error from server: code=2200 [Invalid query] message="Undefined column name dept_name"
cqlsh:test> insert into dept values(1001,'Accounts','Mumbai');
SyntaxException: line 1:17 no viable alternative at input 'values' (insert into [dept] values...)
cqlsh:test> insert into dept(dept_id,dept_name,dept_loc);
SyntaxException: line 1:44 mismatched input ';' expecting K_VALUES (...into dept(dept_id,dept_name,dept_loc)[;])
cqlsh:test> insert into dept(dept_id,dept_name,dept_loc) values(10,'accounts','mumbai');
InvalidRequest: Error from server: code=2200 [Invalid query] message="Undefined column name dept_name"
cqlsh:test> insert into dept(dept_id,dept_nmae,dept_loc) values(10,'accounts','mumbai');
cqlsh:test> select * from dept;

 dept_id | dept_loc | dept_nmae
---------+----------+-----------
      10 |   mumbai |  accounts

(1 rows)
cqlsh:test> insert into dept(dept_id,dept_nmae,dept_loc) values(20,'sales','delhi');
cqlsh:test> insert into dept(dept_id,dept_nmae,dept_loc) values(30,'HR','chennai');
cqlsh:test> select * from dept;

 dept_id | dept_loc | dept_nmae
---------+----------+-----------
      10 |   mumbai |  accounts
      30 |  chennai |        HR
      20 |    delhi |     sales

(3 rows)
cqlsh:test> insert into dept(dept_id,dept_nmae,dept_loc) values(40,'marketing','kerala');
cqlsh:test> select * from dept;

 dept_id | dept_loc | dept_nmae
---------+----------+-----------
      10 |   mumbai |  accounts
      30 |  chennai |        HR
      20 |    delhi |     sales
      40 |   kerala | marketing

(4 rows)
cqlsh:test>
```

```
(4 rows)
cqlsh:test> insert into emp(emp_id,emp_city,emp_name,emp_phone,emp_sal) values(1,'mumbai','shreya',9320186120,40000)
       ... ;
cqlsh:test> insert into emp(emp_id,emp_city,emp_name,emp_phone,emp_sal) values(1,'mumbai','shreya',9320186120,40000) ;
cqlsh:test> insert into emp(emp_id,emp_city,emp_name,emp_phone,emp_sal) values(2,'delhi','sushma',9376745680,50000) ;
cqlsh:test> insert into emp(emp_id,emp_city,emp_name,emp_phone,emp_sal) values(3,'chennai','rachana',9876874680,60000) ;
cqlsh:test> insert into emp(emp_id,emp_city,emp_name,emp_phone,emp_sal) values(4,'kerala','soni',7656257899,30000) ;
cqlsh:test> insert into emp(emp_id,emp_city,emp_name,emp_phone,emp_sal) values(5,'bhopal','priya',9908765869,70000) ;
cqlsh:test> select * from emp;

 emp_id | emp_city | emp_name | emp_phone  | emp_sal
--------+----------+----------+------------+---------
      5 |   bhopal |    priya | 9908765869 |   70000
      1 |   mumbai |   shreya | 9320186120 |   40000
      2 |    delhi |   sushma | 9376745680 |   50000
      4 |   kerala |     soni | 7656257899 |   30000
      3 |  chennai |  rachana | 9876874680 |   60000

(5 rows)
cqlsh:test>
```

```
cqlsh:test> CREATE TABLE emp (
        ... emp_id int PRIMARY KEY,
        ... emp_name text,
        ... emp_city text,
        ... emp_sal varint,
        ... emp_phone varint,
        ... );
cqlsh:test> select * from emp;

 emp_id | emp_city | emp_name | emp_phone | emp_sal
--------+----------+----------+-----------+---------

(0 rows)
```

Command Prompt - cqlsh

```
(5 rows)
cqlsh:test> create table student(std_id int PRIMARY KEY,std_name text,std_grade text);
cqlsh:test> insert into student(std_id,std_name,std_grade) values(1,'abhijit','O') ;
cqlsh:test> insert into student(std_id,std_name,std_grade) values(2,'pratik','A') ;
cqlsh:test> insert into student(std_id,std_name,std_grade) values(3,'rahul','B') ;
cqlsh:test> insert into student(std_id,std_name,std_grade) values(4,'yash','C') ;
cqlsh:test> insert into student(std_id,std_name,std_grade) values(5,'chinmay','D') ;
cqlsh:test> select * from student;

 std_id | std_grade | std_name
--------+-----------+----------
      5 |         D |  chinmay
      1 |         O |  abhijit
      2 |         A |   pratik
      4 |         C |     yash
      3 |         B |    rahul

(5 rows)
cqlsh:test>
```

Command Prompt - cqlsh

```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\acer>cd ..

C:\Users>cd /

C:\>cd apache-cassandra-3.11.14\bin

C:\apache-cassandra-3.11.14\bin>cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.14 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> create keyspace test;
SyntaxException: line 1:20 mismatched input ';' expecting K_WITH (create keyspace test[;])
cqlsh> CREATE KEYSPACE tutorialspoint
   ... WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};

Warnings :
Your replication factor 3 for keyspace tutorialspoint is higher than the number of nodes 1

cqlsh> CREATE KEYSPACE tutorialspoint WITH replication = {'class':'SimpleStrategy', 'replication_factor' :1};
AlreadyExists: Keyspace 'tutorialspoint' already exists
cqlsh> CREATE KEYSPACE test WITH replication = {'class':'SimpleStrategy', 'replication_factor' :1};
cqlsh> Use test;
cqlsh:test> CREATE TABLE emp(
        ... emp_id int PRIMARY KEY,
        ... emp_nmae text,
        ... emp_city text;
SyntaxException: line 4:13 mismatched input ';' expecting ')'
cqlsh:test> CREATE TABLE emp (
        ... emp_id int PRIMARY KEY,
        ... emp_name text,
        ... emp_city text,
        ... emp_sal varint,
        ... emp_phone varint,
        ... );
cqlsh:test> select * from emp;

 emp_id | emp_city | emp_name | emp_phone | emp_sal
```

Command Prompt - cqlsh

```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\acer>cd
C:\Users\acer

C:\Users\acer>cd\

C:\>cd apache-cassandra-3.11.14\bin

C:\apache-cassandra-3.11.14\bin>cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.14 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> use test
   ... ;
cqlsh:test> select * from dept
        ... ;

 dept_id | dept_loc | dept_nmae
---------+----------+-----------
      10 |   mumbai |  accounts
      30 |  chennai |        HR
      20 |    delhi |     sales
      40 |   kerala | marketing

(4 rows)
cqlsh:test> update dept set dept_nmae='finance' where dept_id=10;
cqlsh:test> select * from dept ;

 dept_id | dept_loc | dept_nmae
---------+----------+-----------
      10 |   mumbai |   finance
      30 |  chennai |        HR
      20 |    delhi |     sales
      40 |   kerala | marketing

(4 rows)
cqlsh:test>
```

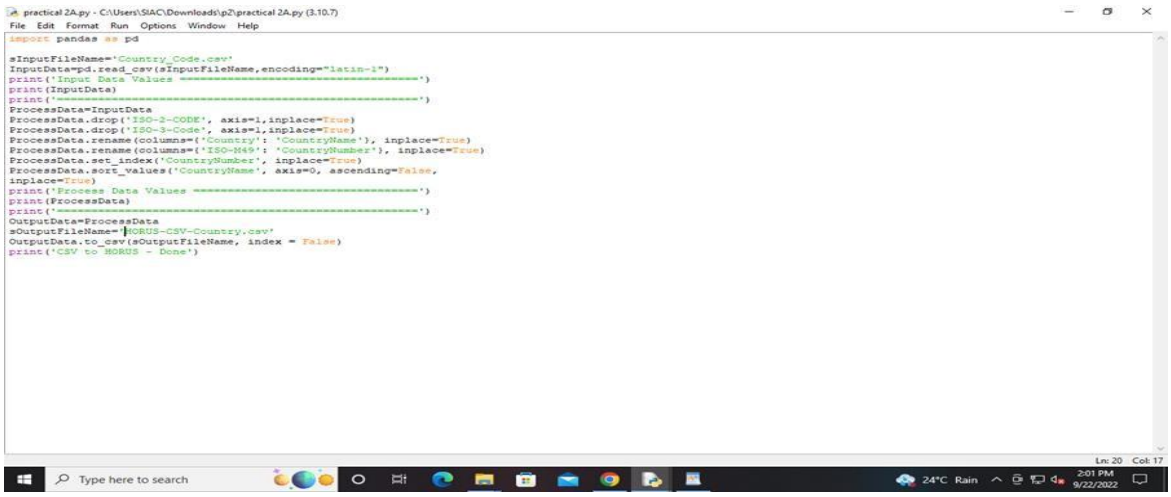
```
(4 rows)
cqlsh:test> delete from dept where dept_id=10;
cqlsh:test> select * from dept ;

 dept_id | dept_loc | dept_nmae
---------+----------+-----------
      30 |  chennai |        HR
      20 |    delhi |     sales
      40 |   kerala | marketing

(3 rows)
cqlsh:test>
```

Practical:2

Conversion from different formats to HOURS format.

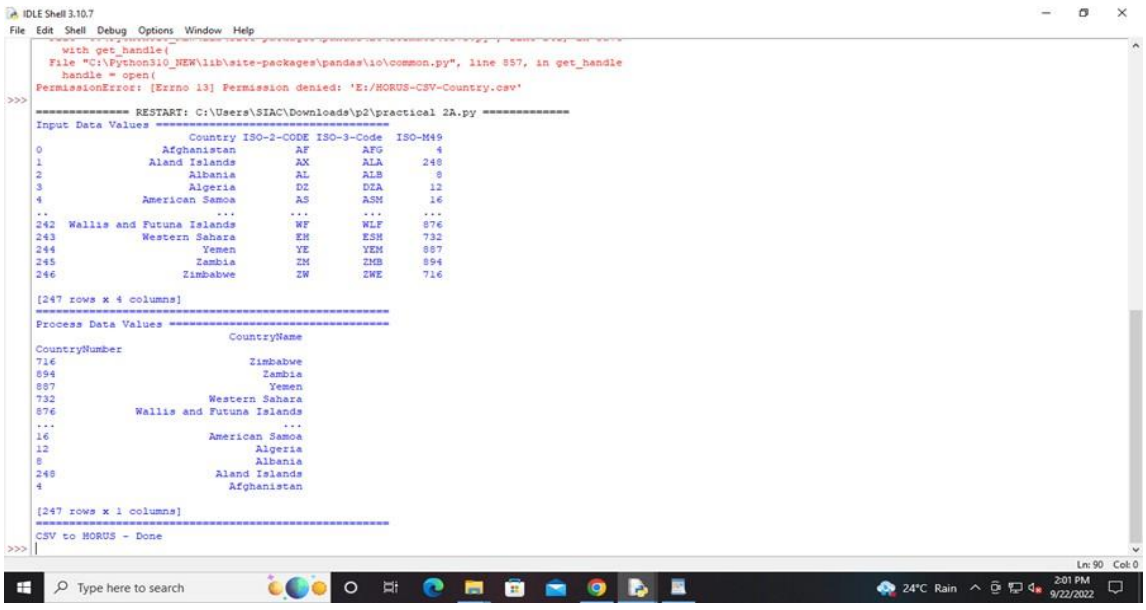
A. Text delimited csv format.

Code:



```
import pandas as pd

sInputFileName='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='HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
```

Output:



```
    with get_handle(
  File "C:\Python310_NEW\lib\site-packages\pandas\io\common.py", line 857, in get_handle
    handle = open(
PermissionError: [Errno 13] Permission denied: 'E:/HORUS-CSV-Country.csv'
>>>
============= RESTART: C:\Users\SIAC\Downloads\p2\practical 2A.py =============
Input Data Values =====================================
                  Country ISO-2-CODE ISO-3-Code  ISO-M49
0             Afghanistan         AF        AFG        4
1           Aland Islands         AX        ALA      248
2                 Albania         AL        ALB        8
3                 Algeria         DZ        DZA       12
4          American Samoa         AS        ASM       16
..                    ...        ...        ...      ...
242  Wallis and Futuna Islands    WF        WLF      876
243         Western Sahara        EH        ESH      732
244                 Yemen         YE        YEM      887
245                Zambia         ZM        ZMB      894
246              Zimbabwe         ZW        ZWE      716

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

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

B. XML.

Code:

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

```
all_records.append(record)
    return pd.DataFrame(all_records)
# Input Agreement
sInputFileName='E:/Softwares_2021-22/MSCIT-II/practical-data-science-master/practical-data-science-master/VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()
print('=================================================')
print('Input Data Values ===============================')
print('=================================================')
print(InputData)
print('=================================================')
# Processing Rules
ProcessDataXML=InputData
# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False,
inplace=True)
print('=================================================')
print('Process Data Values =============================')
print('=================================================')
print(ProcessData)
print('=================================================')
```

```
# Output Agreement
OutputData=ProcessData
sOutputFileName='E:/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('============================================================')
print('XML to HORUS - Done')
print('============================================================')
# Utility done
```

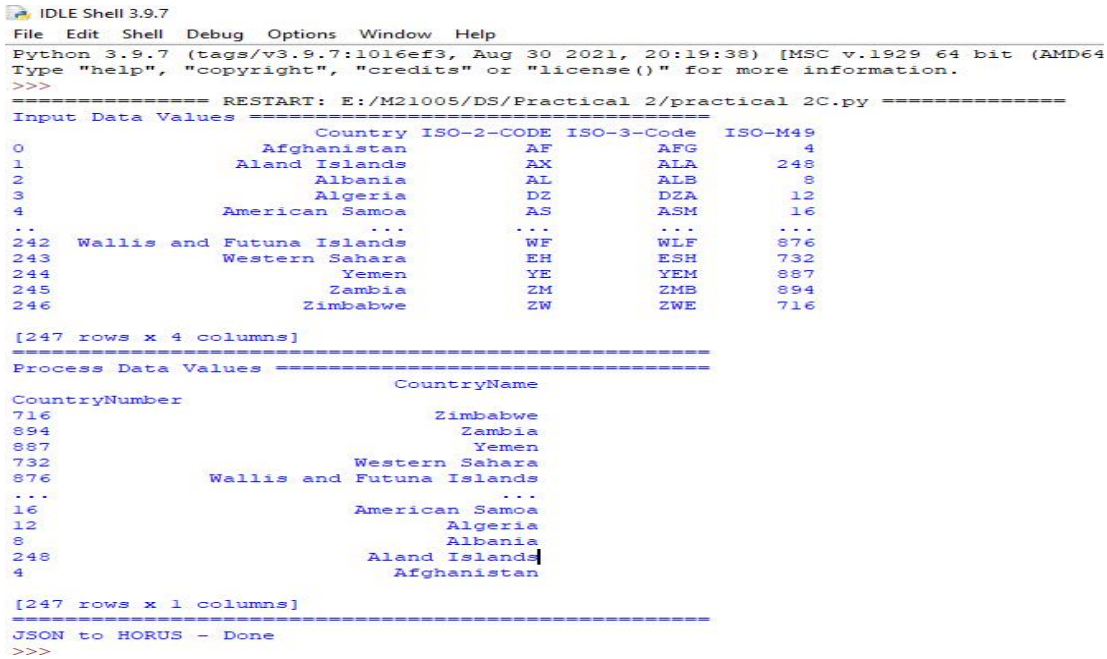
Output:

C. JSON.

Code:

```
import pandas as pd
sInputFileName='C:/Users/SIALAB01PC12/Downloads/Country_Code.json'
InputData=pd.read_json(sInputFileName,
                       orient='index',
                       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='E:\M21005\DS\Practical 2\HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('JSON to HORUS - Done')
```

Output:



```
IDLE Shell 3.9.7
File  Edit  Shell  Debug  Options  Window  Help
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: E:/M21005/DS/Practical 2/practical 2C.py ==============
Input Data Values ======================================
                      Country ISO-2-CODE ISO-3-Code  ISO-M49
0                 Afghanistan          AF         AFG        4
1               Aland Islands          AX         ALA      248
2                     Albania          AL         ALB        8
3                     Algeria          DZ         DZA       12
4              American Samoa          AS         ASM       16
..                        ...         ...         ...      ...
242  Wallis and Futuna Islands          WF         WLF      876
243             Western Sahara          EH         ESH      732
244                      Yemen          YE         YEM      887
245                     Zambia          ZM         ZMB      894
246                   Zimbabwe          ZW         ZWE      716

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

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

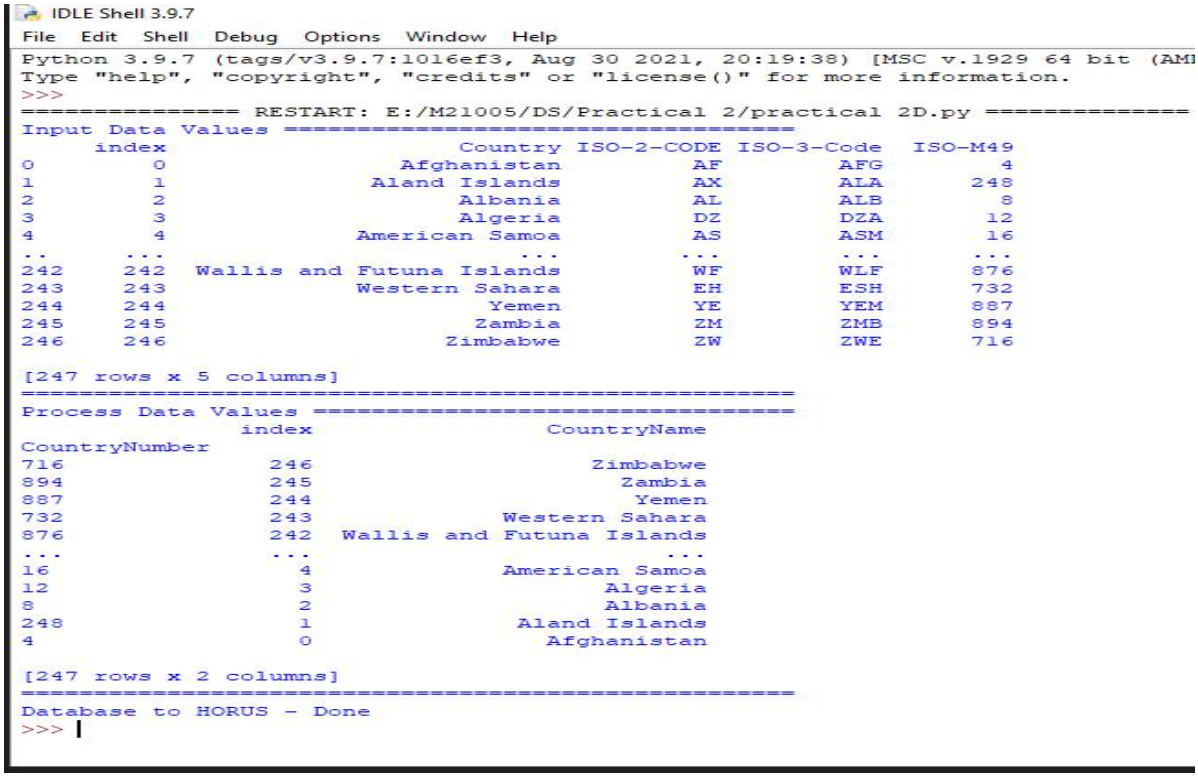
D. MySQL Database.

Code:

```
# Utility Start Database to HORUS
# Standard Tools
import pandas as pd
import sqlite3 as sq
# Input Agreement
sInputFileName='C:/Users/SIALAB01PC12/Downloads/utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)
print('Input Data Values ===================================')
print(InputData)
print('=============================================================')
# Processing Rules
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values ================================')
print(ProcessData)
```

```
print('==================================================')
# Output Agreement
OutputData=ProcessData
sOutputFileName='E:/M21005/DS/Practical 2/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('Database to HORUS - Done')
```

Output:



E. Video.

Code:

frame.py - D:/New folder/mk/frame.py (3.10.7)
File Edit Format Run Options Window Help

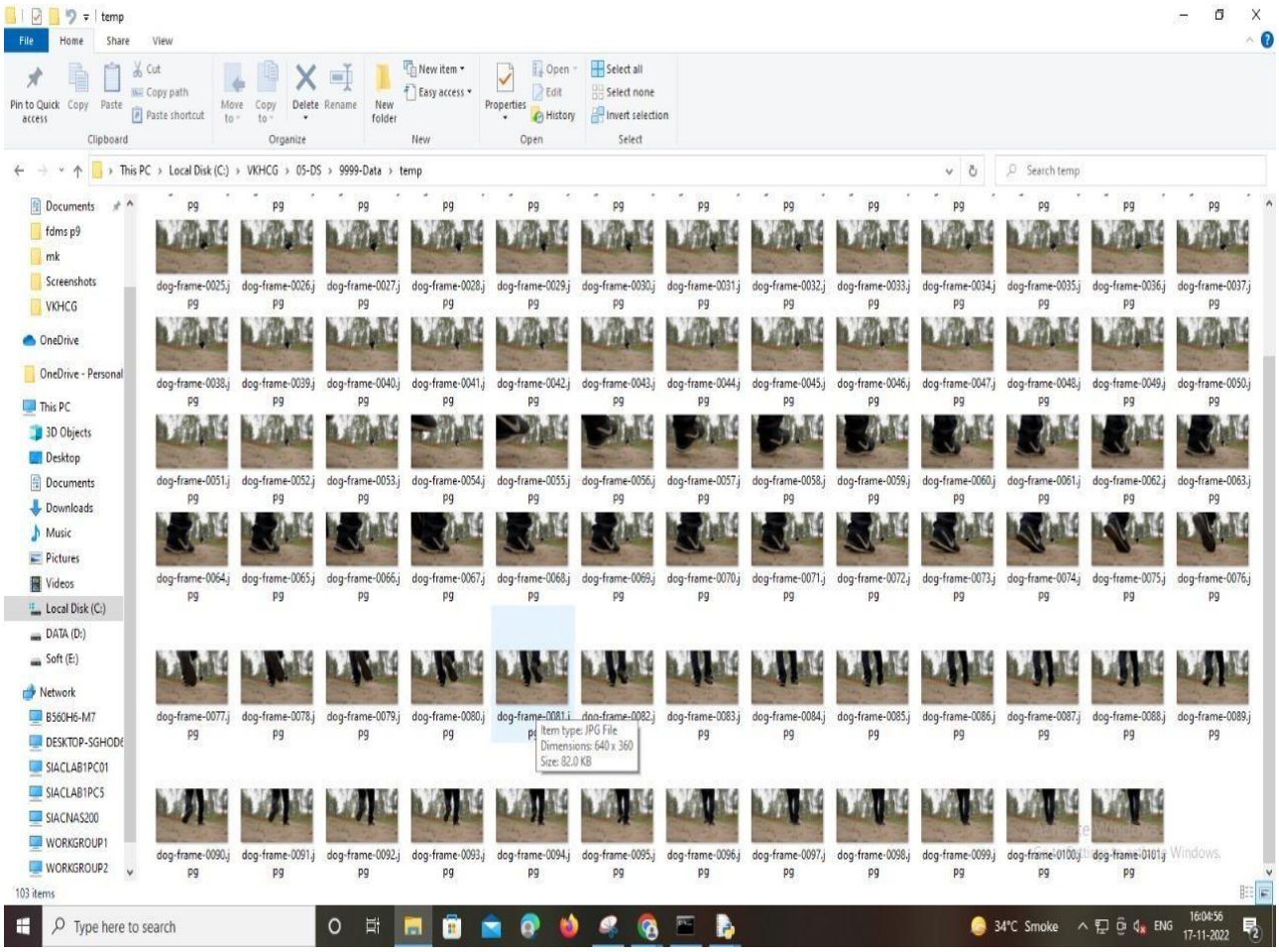
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
# Input Agreement ============================================
sDataBaseDir='C:/VKHCG/05-DS/9999-Data/Dog.mp4'
f=0
for file in os.listdir(sDataBaseDir):
    if file.endswith(".jpg"):
        f += 1
        sInputFileName=os.path.join(sDataBaseDir, file)
        print('Process : ', sInputFileName)
        InputData = imread(sInputFileName, flatten=False, mode='RGBA')
        print('Input Data Values ===================================')
        print('X: ',InputData.shape[0])
        print('Y: ',InputData.shape[1])
        print('RGBA: ', InputData.shape[2])
        print('=====================================================')
        # Processing Rules ============================================
        ProcessRawData=InputData.flatten()
        y=InputData.shape[2] + 2
        x=int(ProcessRawData.shape[0]/y)
        ProcessFrameData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
        ProcessFrameData['Frame']=file
        print('=====================================================')
        print('Process Data Values ================================')
        print('=====================================================')
        plt.imshow(InputData)
```

Type here to search

```
if f > 0:
    sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha','FrameName']
    ProcessData.columns=sColumns
    print('=====================================================')
    ProcessFrameData.index.names =['ID']
    print('Rows: ',ProcessData.shape[0])
    print('Columns :',ProcessData.shape[1])
    print('=====================================================')
    # Output Agreement ============================================
    OutputData=ProcessData
    print('Storing File')
    sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Movie-Frame.csv'
    OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Processed ; ', f,' frames')
print('=====================================================')
print('Movie to HORUS - Done')
print('=====================================================')
```

Type here to search

Output:

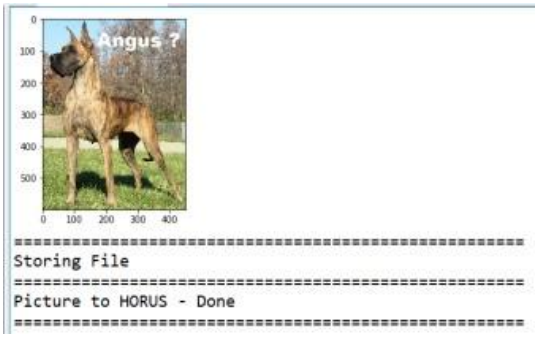


F. Picture (JPEG) to HORUS format

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

```
InputData = imread(sInputFileName, flatten=False, mode='RGBA')

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

print('X: ',InputData.shape[0])

print('Y: ',InputData.shape[1])

print('RGBA: ', InputData.shape[2])
print('=====================================================')

# Processing Rules ============================================
ProcessRawData=InputData.flatten()

y=InputData.shape[2] + 2

x=int(ProcessRawData.shape[0]/y)

ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))

sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']

ProcessData.columns=sColumns ProcessData.index.names =['ID'] print('Rows:
',ProcessData.shape[0]) print('Columns :',ProcessData.shape[1])
print('=====================================================')

print('Process Data Values ================================')
print('=====================================================')

plt.imshow(InputData) plt.show()
print('=====================================================')

# Output Agreement ============================================

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

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

OUTPUT:



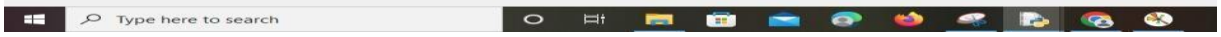
G. Audio.

Code:

audio.py - D:/New folder/mk/audio.py (3.10.7)

File Edit Format Run Options Window Help

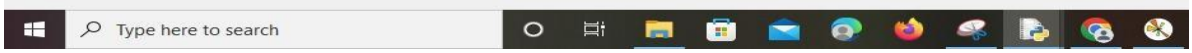
```
from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
def show_info(aname, a,r):
    print ('----------------')
    print ("Audio:", aname)
    print ('----------------')
    print ("Rate:", r)
    print ('----------------')
    print ("shape:", a.shape)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



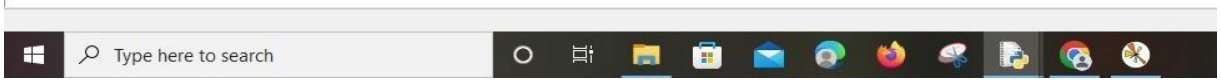
audio.py - D:/New folder/mk/audio.py (3.10.7)

File Edit Format Run Options Window Help

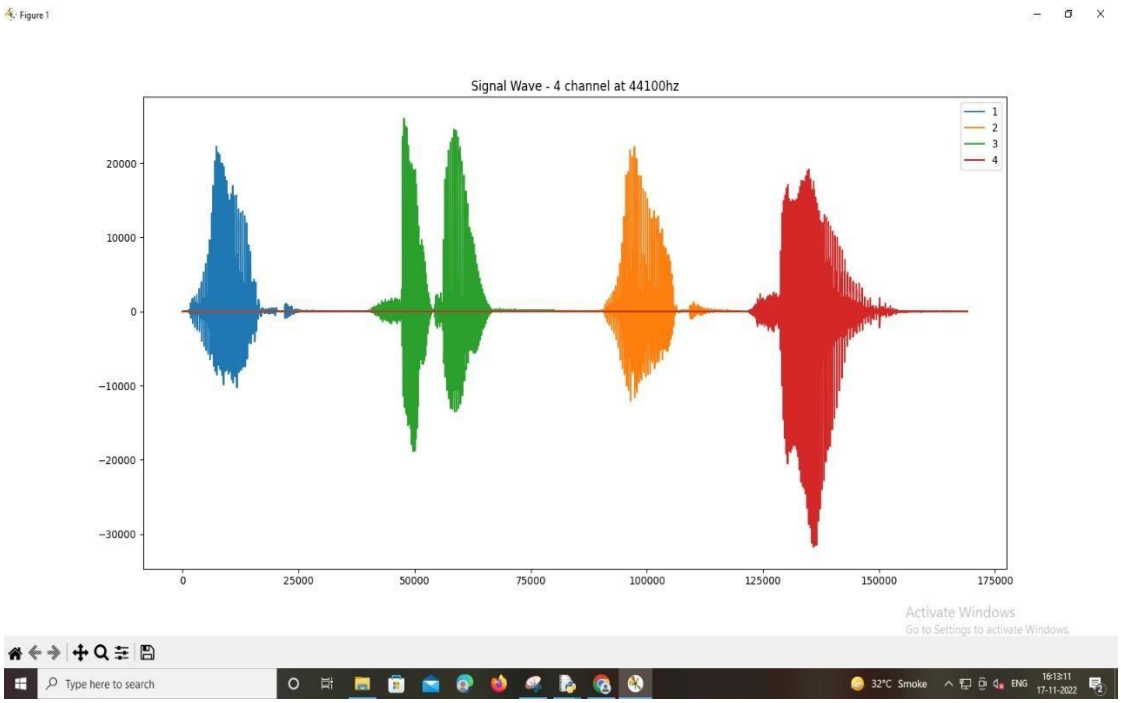
```
print('Processing : ', sInputFileName)
print('=============================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
OutputData=ProcessData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



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



Output:

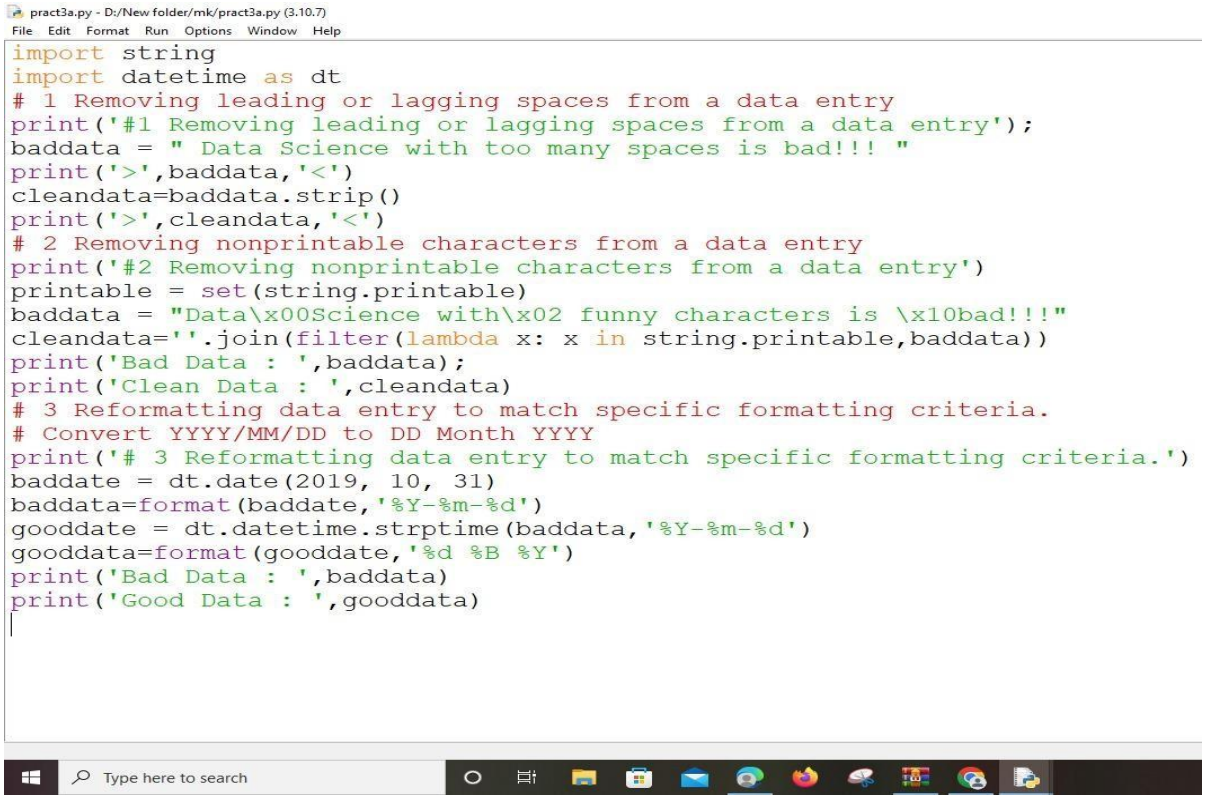


Practical:3

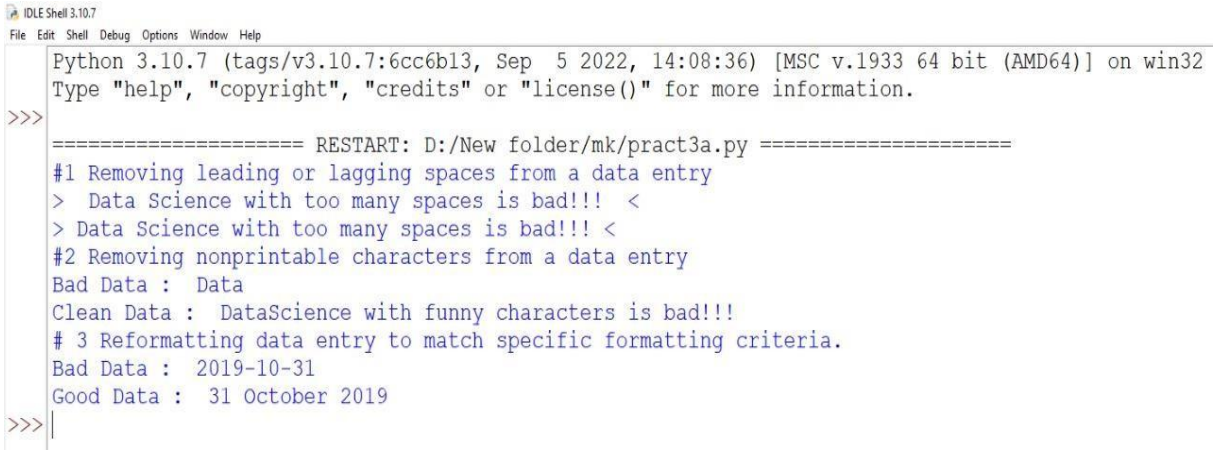
Utilities and Auditing.

A.Fixers Utilities.

Code:



Output:

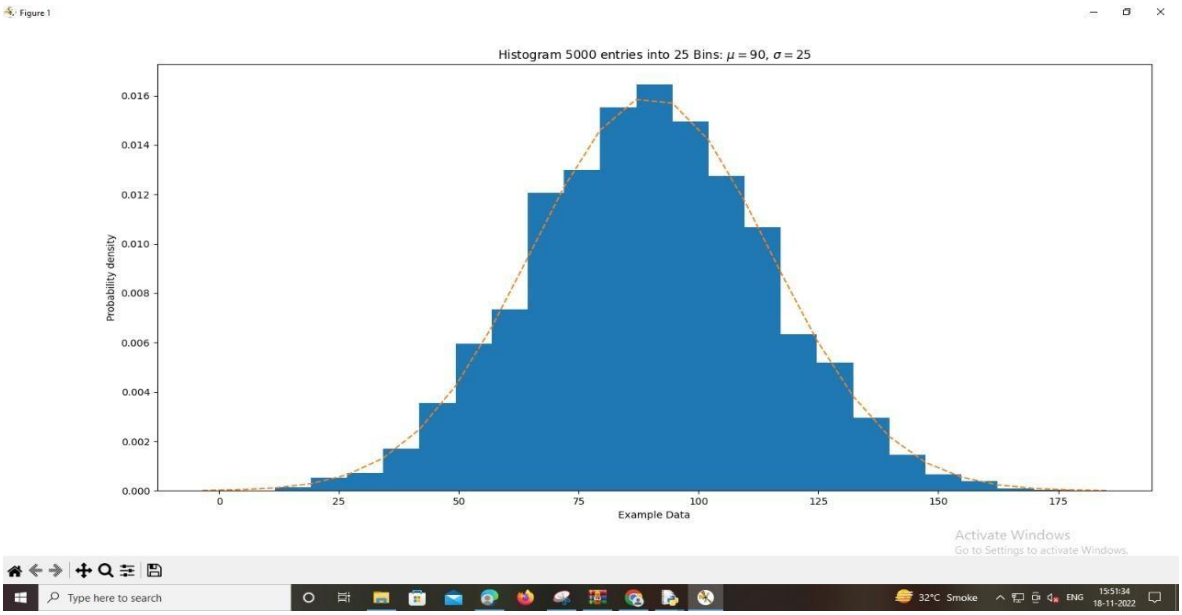


B. Data Binning or Bucketing.

Code:

```
3B.py - D:/New folder/mk/3B.py (3.10.7)
File  Edit  Format  Run  Options  Window  Help
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='
ax.set_title(sTitle)
fig.tight_layout()
sPathFig='D:/New folder/mk/practical-data-science-master/VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png'
fig.savefig(sPathFig)
plt.show()
```


Output:



C. Logging.

Code:

```
import sys
import os
import logging
import uuid
import shutil
import time
Base='c:\ds\ds\practical-data-science-master\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']
#You can now build the loops to perform a basic logging run.
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)
#You set up logging to file, as follows:
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(name)-12s %(levelname)-
8s%(message)s',datefmt='%m-%d %H:%M',filename=sLogFile,filemode='w')
#You define a handler, which writes all messages to sys.stderr.
console = logging.StreamHandler()
console.setLevel(logging.INFO)
#You set a format for console use.
 formatter = logging.Formatter('%(name)-12s: %(levelname)-8s%(message)s')
#You activate the handler to use this format.
console.setFormatter(formatter)
#Now, add the handler to the root logger.
logging.getLogger('').addHandler(console)
#Test your root logging.
logging.info('Practical Data Science is fun!.')
#Test all the other levels.
 for sLevel in sLevels:
sApp='Application-'+ 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:


IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/New folder/mk/log.py ==========
Set up: c:\ds\ds\practical-data-science-master\VKHCG/01-Vermeulen/06-Report/Logging/Logging_005dabfb-d0d8-4c72-935a-1d690b2046d5.log
root : INFO Practical Data Science is fun!.
Application-01-Vermeulen-06-Report-info: INFO Practical Data Science logged information message.
Application-01-Vermeulen-06-Report-warning: WARNING Practical Data Science logged a warning message.
Application-01-Vermeulen-06-Report-error: ERROR Practical Data Science logged an error message.
Set up: c:\ds\ds\practical-data-science-master\VKHCG/02-Krennwallner/06-Report/Logging/Logging_5160e455-2296-4eba-9c1f-2b0a4ba6fa09.log
root : INFO Practical Data Science is fun!.
Application-02-Krennwallner-06-Report-info: INFO Practical Data Science logged information message.
Application-02-Krennwallner-06-Report-warning: WARNING Practical Data Science logged a warning message.
Application-02-Krennwallner-06-Report-error: ERROR Practical Data Science logged an error message.
Set up: c:\ds\ds\practical-data-science-master\VKHCG/03-Hillman/06-Report/Logging/Logging_43861ba8-a884-43a1-b62d-37399a225dff.log

D. Outter dedection

Code:

```
# -*- coding: utf-8 -*-
###################################################################### 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) &
(AllData.Latitude<=UpperBound)] print(OutliersNot)
######################################################################
```

OUTPUT:

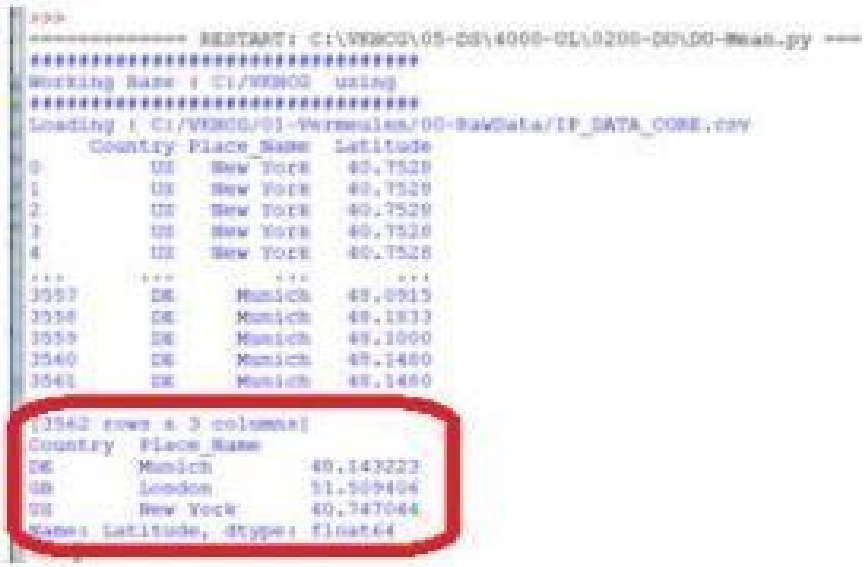
```
========== RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py ==========
################################
Working Base : C:/VKHCG
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
All Data
     Country Place_Name  Latitude
1910      GB     London   51.5130


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

[1502 rows x 3 columns]
Outliers
Higher than  51.51263550786781
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
1916      GB     London   51.5491
1919      GB     London   51.5161
1920      GB     London   51.5198
1921      GB     London   51.5198
1923      GB     London   51.5237
1924      GB     London   51.5237
1925      GB     London   51.5237
1926      GB     London   51.5237
1927      GB     London   51.5232
3436      GB     London   51.5163
3438      GB     London   51.5136
Lower than  51.50617687562166
     Country Place_Name  Latitude
1915      GB     London   51.4739
Not Outliers
     Country Place_Name  Latitude
1917      GB     London   51.5085
1918      GB     London   51.5085
1922      GB     London   51.5085
1928      GB     London   51.5085
1929      GB     London   51.5085
...      ...        ...       ...
3432      GB     London   51.5092
3433      GB     London   51.5092
3434      GB     London   51.5092
3435      GB     London   51.5092
3437      GB     London   51.5085
[1485 rows x 3 columns]
```

E.Averaging of Data

Code:

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

OutputFileName='Retrieve_Router_Location.csv'

Base='C:/VKHCG' print('##################################')

print('Working Base :',Base, ' using ')

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

sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName

print('Loading :',sFileName)

IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,

usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
AllData=IP_DATA_ALL[['Country', 'Place_Name','Latitude']]

 print(AllData)

MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()

print(MeanData) #####################################################################
```

OUTPUT:



Practical:4

Retrieving Data.

A. Loading IP_DATA_ALL:

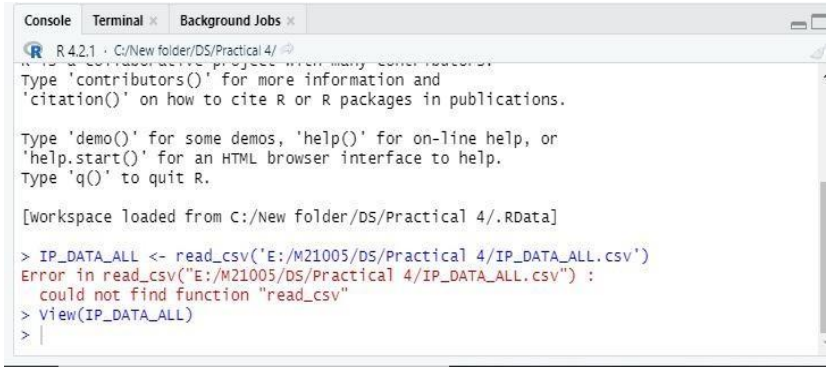
Code:

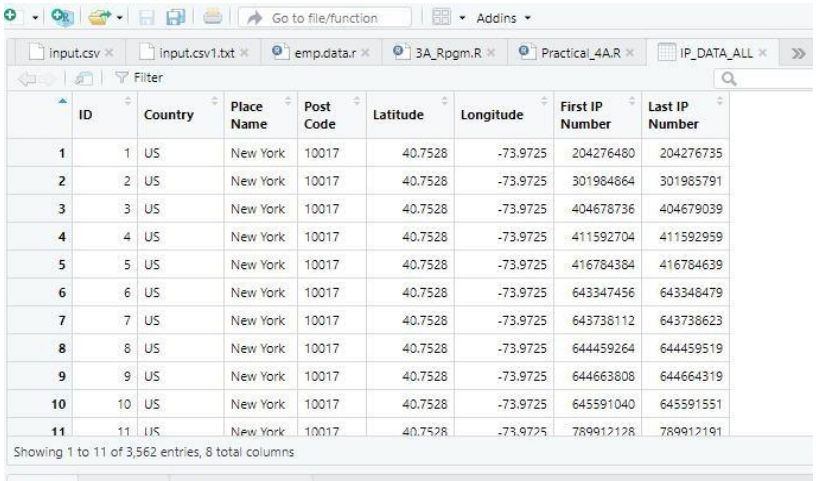
```
library(readr)
IP_DATA_ALL <- read_csv('E:/M21005/DS/Practical 4/IP_DATA_ALL.csv')
View(IP_DATA_ALL)
spec(IP_DATA_ALL)
library(tibble)
set_tidy_names(IP_DATA_ALL,syntactic = TRUE,quiet = FALSE)
IP_DATA_ALL_FIX = set_tidy_names(IP_DATA_ALL,syntactic = TRUE,quiet =
TRUE)
sapply(IP_DATA_ALL_FIX,typeof)
write.csv(IP_DATA_ALL_FIX,'E:/M21005/DS/Practical
4/IP_DATA_ALL_FIX.csv')
setwd("E:/M21005/DS/Practical 4/")
library(data.table)
IP_DATA_ALL_with_ID = rowid_to_column(IP_DATA_ALL_FIX, var = "RowID")
View(IP_DATA_ALL_with_ID)
write.csv(IP_DATA_ALL_with_ID,'IP_DATA_ALL_with_ID.csv')
sapply(IP_DATA_ALL_with_ID, typeof)
library(data.table)
hist_country = data.table(Country =
unique(IP_DATA_ALL_with_ID[is.na(IP_DATA_ALL_with_ID['Country']) ==
0,]$Country))
setorder(hist_country,'Country')
hist_country_with_id = rowid_to_column(hist_country, var = "RowIDCountry")
View(hist_country_with_id)
IP_DATA_COUNTRY_FREQ = data.table(with(IP_DATA_ALL_with_ID,
table(Country)))
View(IP_DATA_COUNTRY_FREQ)
write.csv(IP_DATA_COUNTRY_FREQ,'IP_DATA_COUNTRY_FREQ.csv')

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

```
IP_DATA_Latitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,table(Latitude)
))
View(IP_DATA_Latitude_FREQ)
write.csv(IP_DATA_Latitude_FREQ,'IP_DATA_Latitude_FREQ.csv')
hist_longitude =data.table(Longitude=unique(IP_DATA_ALL_with_ID
[is.na(IP_DATA_ALL_with_ID ['Longitude']) == 0, ]$Longitude))
setkeyv(hist_longitude,  'Longitude') setorder(hist_longitude,'Longitude')
hist_longitude_with_id=rowid_to_column(hist_longitude, var = "RowID")
View(hist_longitude_with_id)
IP_DATA_Longitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,table(Longit
ude)))
View(IP_DATA_Longitude_FREQ)
write.csv(IP_DATA_Longitude_FREQ,'IP_DATA_Longitude_FREQ.csv')
#minimum values
min(hist_country$Country)
sapply(hist_country[,'Country'], min, na.rm=TRUE)
sapply(hist_latitude_with_id[,'Latitude'], min, na.rm=TRUE)
sapply(hist_longitude_with_id[,'Longitude'], min, na.rm=TRUE)
#maximum values
max(hist_country$Country)
sapply(hist_country[,'Country'], max, na.rm=TRUE)
sapply(hist_latitude_with_id[,'Latitude'], max, na.rm=TRUE)
sapply(hist_longitude_with_id[,'Longitude'], max, na.rm=TRUE)
#Mean
sapply(hist_latitude_with_id[,'Latitude'], mean, na.rm=TRUE)
sapply(hist_longitude_with_id[,'Longitude'], mean, na.rm=TRUE)
#Median
sapply(hist_latitude_with_id[,'Latitude'], median, na.rm=TRUE)
sapply(hist_longitude_with_id[,'Longitude'], median, na.rm=TRUE)
#Mode
IP_DATA_COUNTRY_FREQ = data.table(with(IP_DATA_ALL_with_ID,
table(Country)))
setorder(IP_DATA_COUNTRY_FREQ,-N)
IP_DATA_COUNTRY_FREQ[1,'Country']
IP_DATA_Latitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,table(Latitude)
))
setorder(IP_DATA_Latitude_FREQ,-N)
IP_DATA_Latitude_FREQ[1,'Latitude']
IP_DATA_Longitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,table(Longit
ude)))
setorder(IP_DATA_Longitude_FREQ,-N)
IP_DATA_Longitude_FREQ[1,'Longitude']
#Range
sapply(hist_latitude_with_id[,'Latitude'], range, na.rm=TRUE)
sapply(hist_longitude_with_id[,'Longitude'], range, na.rm=TRUE)
```

```
sapply(hist_country_with_id[,'Country'], range, na.rm=TRUE)
#Quartiles
sapply(hist_latitude_with_id[,'Latitude'], quantile, na.rm=TRUE)
sapply(hist_longitude_with_id[,'Longitude'], quantile, na.rm=TRUE)
#Standard Deviation
sapply(hist_latitude_with_id[,'Latitude'], sd, na.rm=TRUE)
sapply(hist_longitude_with_id[,'Longitude'], sd, na.rm=TRUE)
#Skewness
library(e1071)
skewness(hist_latitude_with_id$Latitude, na.rm = FALSE, type = 2)
skewness(hist_longitude_with_id$Longitude,na.rm = FALSE, type = 2)
#Missing or Unknown Values
missing_country=data.table(Country=unique(IP_DATA_ALL_with_ID[is.na(IP_DA
TA_ALL_with_ID ['Country']) == 1, ]))
View(missing_country)
```

Output:





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

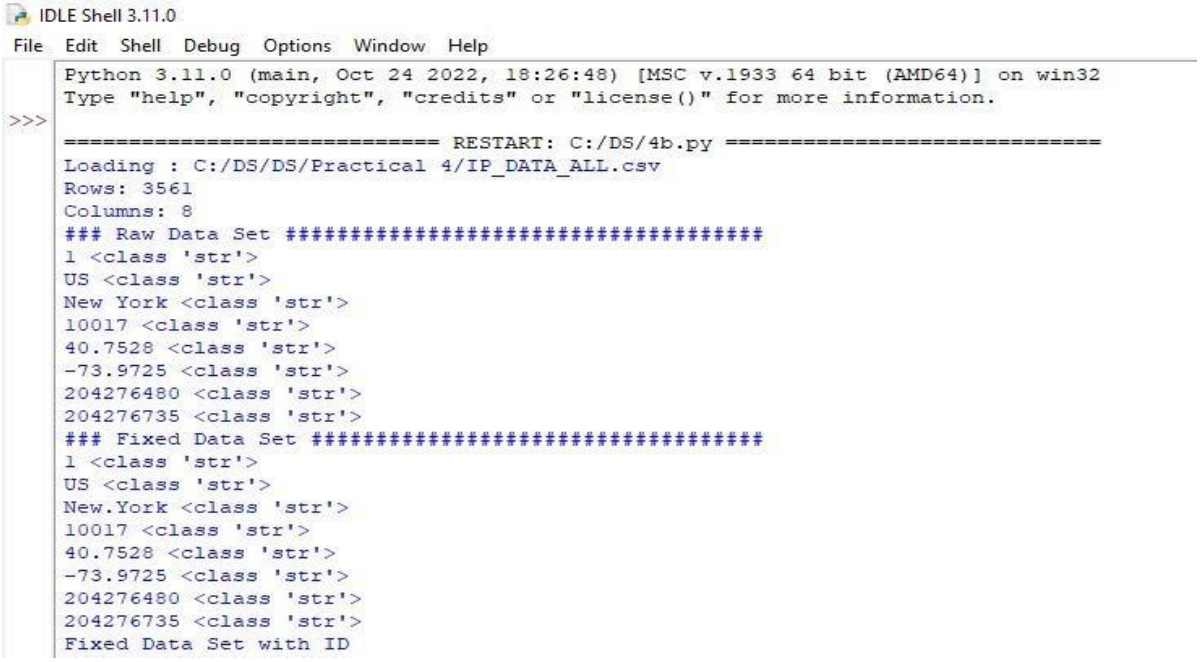
Showing 1 to 11 of 3,562 entries, 8 total columns

B. Program to retrieve different attributes of data.

Code:

```
import sys
import os
import pandas as pd
sFileName='C:/DS/DS/Practical 4/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=1,low_memory=False,encoding="latin-1")
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='/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True)
print('### Done!! ################################################')
```

Output:


IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/DS/4b.py ==========
Loading : C:/DS/DS/Practical 4/IP_DATA_ALL.csv
Rows: 3561
Columns: 8
Raw Data Set
1 <class 'str'>
US <class 'str'>
New York <class 'str'>
10017 <class 'str'>
40.7528 <class 'str'>
-73.9725 <class 'str'>
204276480 <class 'str'>
204276735 <class 'str'>
Fixed Data Set
1 <class 'str'>
US <class 'str'>
New.York <class 'str'>
10017 <class 'str'>
40.7528 <class 'str'>
-73.9725 <class 'str'>
204276480 <class 'str'>
204276735 <class 'str'>
Fixed Data Set with ID

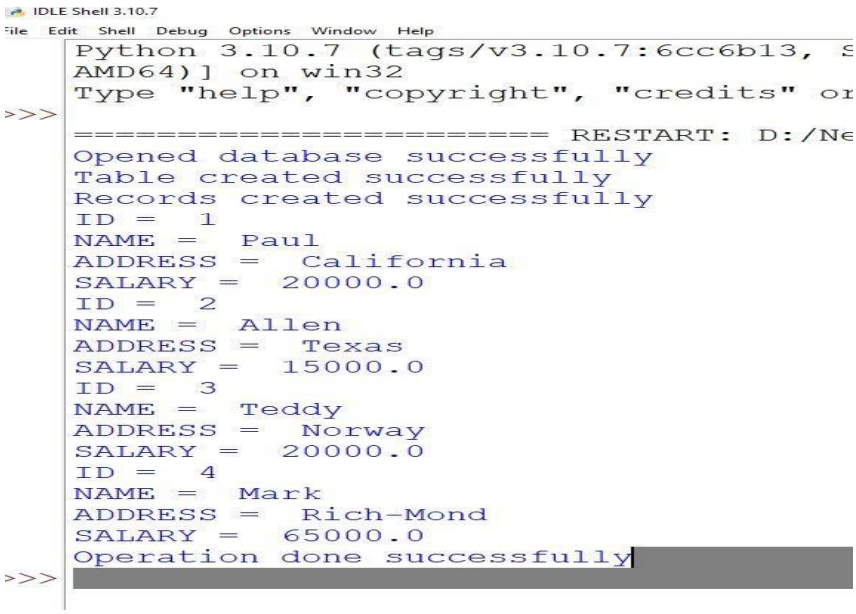
C. Mysql.

Code:

```
import sqlite3
conn = sqlite3.connect('test1.db')
print ("Opened database successfully");
conn.execute("'CREATE TABLE COMPANY
(ID INT PRIMARY KEY NOT NULL,
NAME TEXT NOT NULL,
 AGE INT NOT NULL,
 ADDRESS CHAR(50) NOT NULL,
 SALARY REAL);'")
print("Table created successfully");
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
 VALUES (1, 'Paul', 32, 'California', 20000.00 )");
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
 VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
 VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )");
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
 VALUES (4, 'Mark', 25, 'Rich-Mond', 65000.00 )");
conn.commit()
print("Records created successfully");
cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
 print ("ID = ",row[0])
 print ("NAME = ",row[1])
 print ("ADDRESS = ",row[2])
```

```
 print ("SALARY = ",row[3])
print("Operation done successfully");
```

Output:

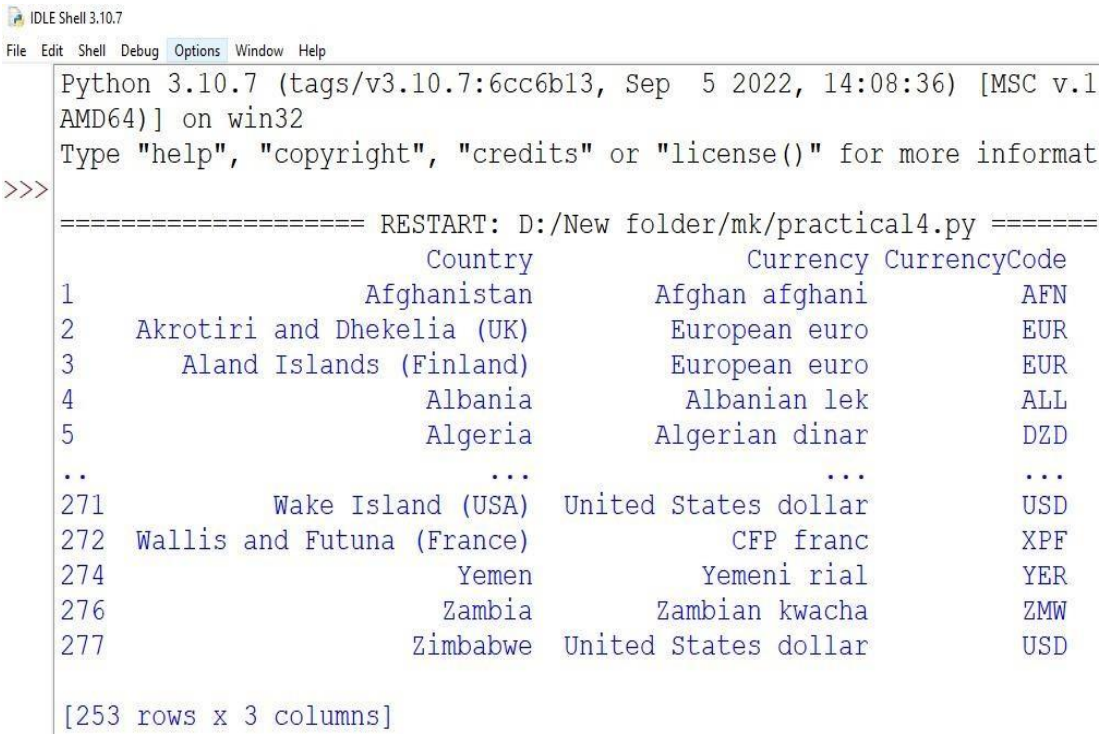


D.Excel.

Code:

```
import os

import pandas as pd

Base='C:/VKHCG'

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

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

CurrencyRawData = pd.read_excel('C:/VKHCG/01-Vermeulen/00-
RawData/Country_Currency.xlsx')

sColumns = ['Country or territory', 'Currency', 'ISO-4217']

CurrencyData = CurrencyRawData[sColumns]

CurrencyData.rename(columns={'Country or territory': 'Country',

'ISO-4217':'CurrencyCode'}, inplace=True)

CurrencyData.dropna(subset=['Currency'],inplace=True)

CurrencyData['Country'] = CurrencyData['Country'].map(lambda x:x.strip())

CurrencyData['Currency'] = CurrencyData['Currency'].map(lambda x:x.strip())

CurrencyData['CurrencyCode'] =CurrencyData['CurrencyCode'].map(lambda x:x.strip())

print(CurrencyData)

sFileName=sFileDir + '/Retrieve-Country-Currency.csv'

CurrencyData.to_csv(sFileName, index = false)
```

Output:


IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informat
>>>
========== RESTART: D:/New folder/mk/practical4.py ==========
Country Currency CurrencyCode
1 Afghanistan Afghan afghani AFN
2 Akrotiri and Dhekelia (UK) European euro EUR
3 Aland Islands (Finland) European euro EUR
4 Albania Albanian lek ALL
5 Algeria Algerian dinar DZD
.. ..
271 Wake Island (USA) United States dollar USD
272 Wallis and Futuna (France) CFP franc XPF
274 Yemen Yemeni rial YER
276 Zambia Zambian kwacha ZMW
277 Zimbabwe United States dollar USD
[253 rows x 3 columns]

Practical:5

Assessing Data.

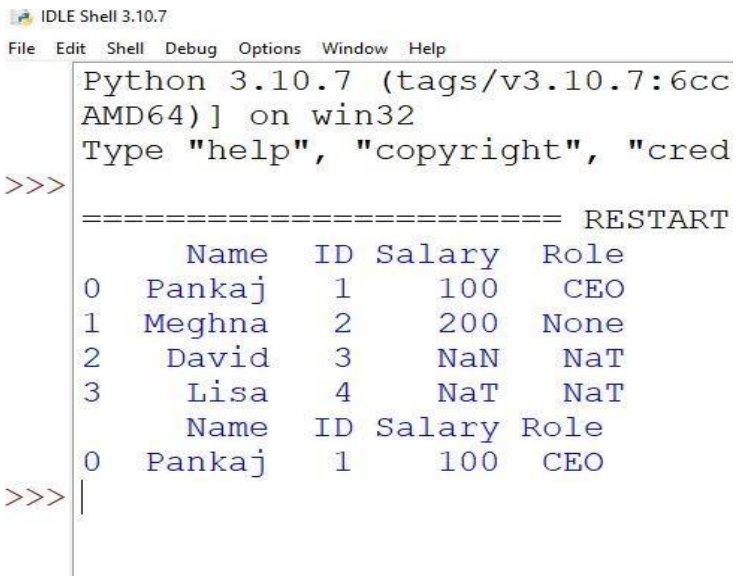
A. Missing Values in Pandas:

1. Drop the Columns Where All Elements Are Missing Values.

Code:

```
import pandas as pd

import numpy as np

d1 = {'Name': ['Pankaj', 'Meghna', 'David', 'Lisa'], 'ID': [1, 2, 3, 4], 'Salary': [100, 200, np.nan, pd.NaT],

    'Role': ['CEO', None, pd.NaT, pd.NaT]}

df = pd.DataFrame(d1)

print(df)

# drop all rows with any NaN and NaT values

df1 = df.dropna()

print(df1)
```

Output:

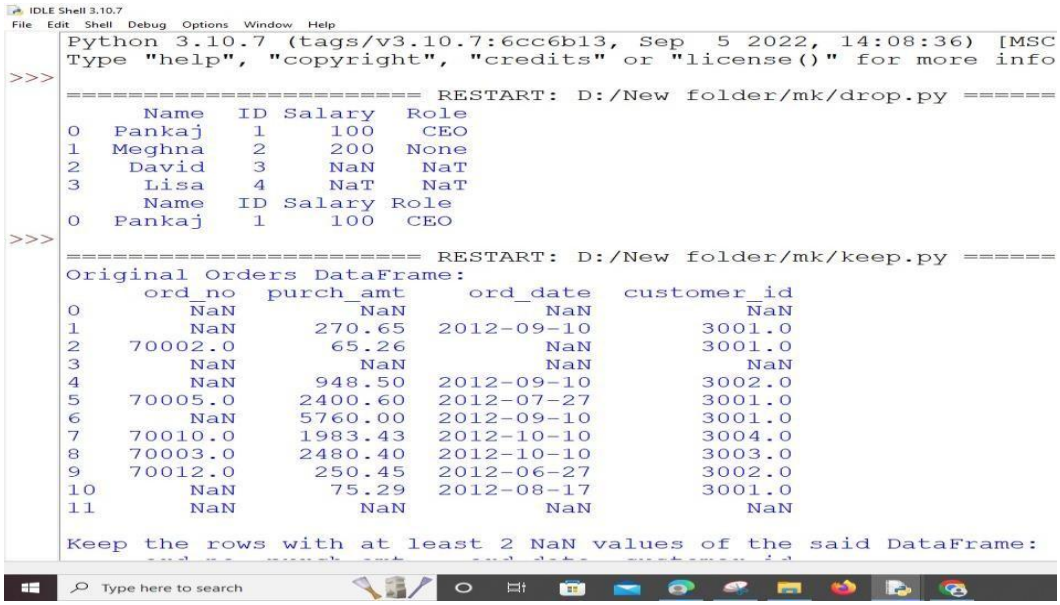


2. Keep Only the Rows That Contain a Maximum of Two Missing Values.

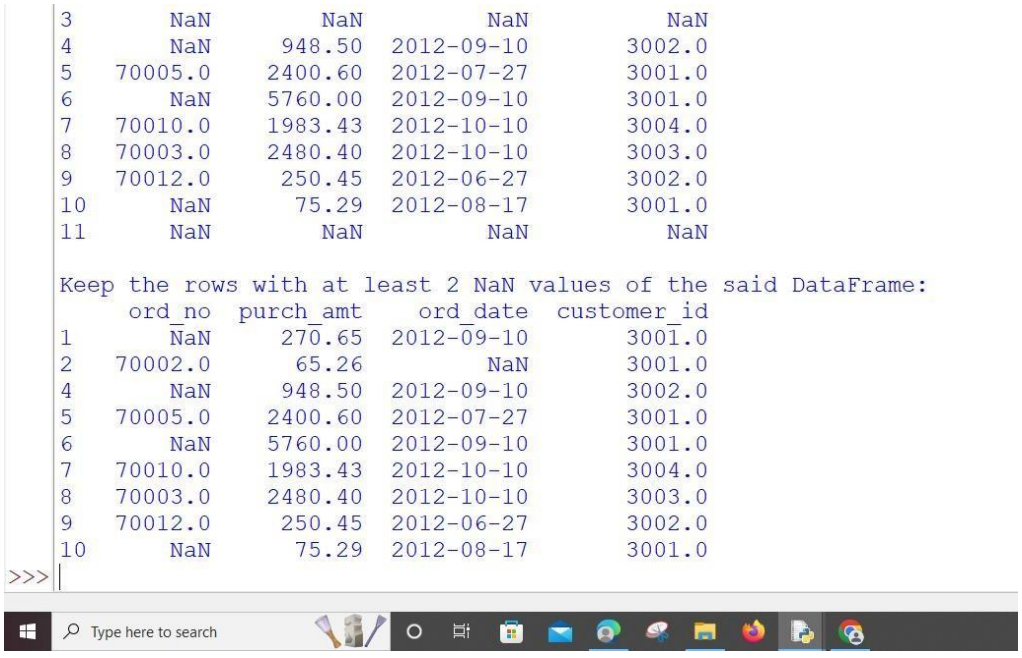
Code:

```
import pandas as pd

import numpy as np

pd.set_option('display.max_rows', None)

#pd.set_option('display.max_columns', None)

df = pd.DataFrame({

'ord_no':[np.nan,np.nan,70002,np.nan,np.nan,70005,np.nan,70010,70003,70012,np.nan,np.nan],

'purch_amt':[np.nan,270.65,65.26,np.nan,948.5,2400.6,5760,1983.43,2480.4,250.45, 75.29,np.nan],

'ord_date': [np.nan,'2012-09-10',np.nan,np.nan,'2012-09-10','2012-07-27','2012-09-10','2012-10-10','2012-10-10','2012-06-27','2012-08-17',np.nan],

'customer_id':[np.nan,3001,3001,np.nan,3002,3001,3001,3004,3003,3002,3001,np.nan]})

print("Original Orders DataFrame:")

print(df)

print("\nKeep the rows with at least 2 NaN values of the said DataFrame:")

result = df.dropna(thresh=2)

print(result)
```

Output:



```
IDLE Shell 3.10.7
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC
Type "help", "copyright", "credits" or "license()" for more info
>>>
========== RESTART: D:/New folder/mk/drop.py ======
     Name  ID Salary  Role
0  Pankaj   1    100   CEO
1  Meghna   2    200  None
2   David   3    NaN   NaT
3    Lisa   4    NaT   NaT
     Name  ID Salary Role
0  Pankaj   1    100  CEO
>>>
========== RESTART: D:/New folder/mk/keep.py ======
Original Orders DataFrame:
     ord_no  purch_amt     ord_date  customer_id
0       NaN        NaN          NaN          NaN
1       NaN     270.65   2012-09-10       3001.0
2   70002.0      65.26          NaN       3001.0
3       NaN        NaN          NaN          NaN
4       NaN     948.50   2012-09-10       3002.0
5   70005.0    2400.60   2012-07-27       3001.0
6       NaN    5760.00   2012-09-10       3001.0
7   70010.0    1983.43   2012-10-10       3004.0
8   70003.0    2480.40   2012-10-10       3003.0
9   70012.0     250.45   2012-06-27       3002.0
10      NaN      75.29   2012-08-17       3001.0
11      NaN        NaN          NaN          NaN

Keep the rows with at least 2 NaN values of the said DataFrame:
```



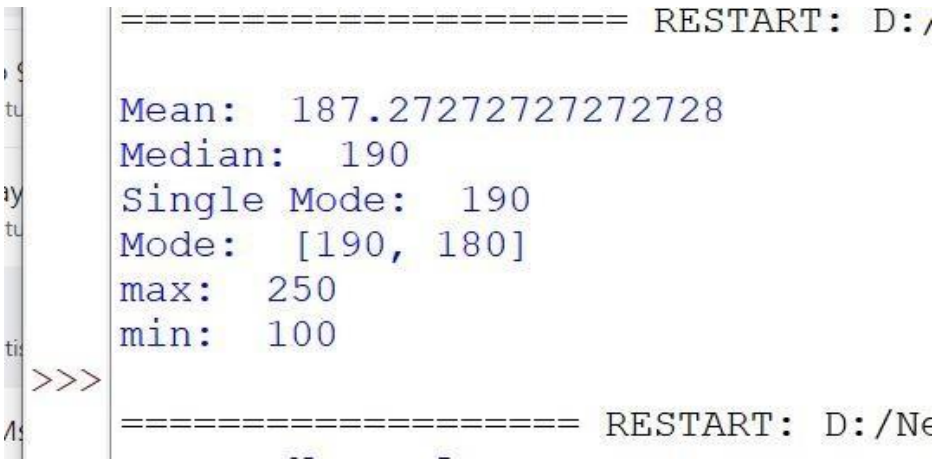
```
3       NaN        NaN          NaN          NaN
4       NaN     948.50   2012-09-10       3002.0
5   70005.0    2400.60   2012-07-27       3001.0
6       NaN    5760.00   2012-09-10       3001.0
7   70010.0    1983.43   2012-10-10       3004.0
8   70003.0    2480.40   2012-10-10       3003.0
9   70012.0     250.45   2012-06-27       3002.0
10      NaN      75.29   2012-08-17       3001.0
11      NaN        NaN          NaN          NaN

Keep the rows with at least 2 NaN values of the said DataFrame:
     ord_no  purch_amt     ord_date  customer_id
1       NaN     270.65   2012-09-10       3001.0
2   70002.0      65.26          NaN       3001.0
4       NaN     948.50   2012-09-10       3002.0
5   70005.0    2400.60   2012-07-27       3001.0
6       NaN    5760.00   2012-09-10       3001.0
7   70010.0    1983.43   2012-10-10       3004.0
8   70003.0    2480.40   2012-10-10       3003.0
9   70012.0     250.45   2012-06-27       3002.0
10      NaN      75.29   2012-08-17       3001.0
>>>
```

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

Code:

```
import statistics
data = [220, 100, 190, 180, 250, 190, 240, 180, 140, 180, 190]
# Finding Mean
print("\nMean: ", statistics.mean(data))
# Finding Median
print("Median: ", statistics.median(data))
# Finding Single Mode
print("Single Mode: ", statistics.mode(data))
# Finding Multiple Modes
print("Mode: ", statistics.multimode(data))
print("max: ",max(data))
print("min: ",min(data))
```

Output:



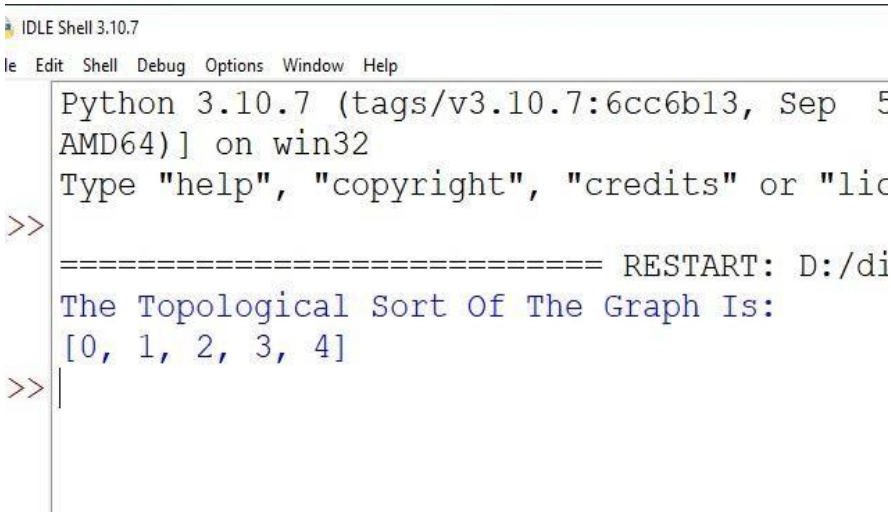
5B.Write a Python / R program to build directed acyclic graph.

Code:

```
from collections import defaultdict
class Graph:
    def _init_(self,n):
self.graph = defaultdict(list)
self.N = n
    def addEdge(self,m,n):
self.graph[m].append
    def sortUtil(self,n,visited,stack):
        visited[n] = True
        for element in self.graph[n]:
            if visited[element] == False:
self.sortUtil(element,visited,stack)
stack.insert(0,n)
    def topologicalSort(self):
        visited = [False]*self.N
        stack =[]
        for element in range(self.N):
            if visited[element] == False:
self.sortUtil(element,visited,stack)
        print(stack)
graph = Graph(5)
graph.addEdge(0,1);
graph.addEdge(0,3);
graph.addEdge(1,2);
graph.addEdge(2,3);
```

```
graph.addEdge(2,4);
graph.addEdge(3,4);
print("The Topological Sort Of The Graph Is: ")
graph.topologicalSort()
```

Output:



5C.Write python/R program to create the network routing diagram from the given data on routers

Code:

```
import sys import os import pandas as pd
#####################################################################
pd.options.mode.chained_assignment = None
#####################################################################

Base='C:/VKHCG'
#####################################################################
print('#################################')

print('Working Base :',Base, ' using Windows')

print('#################################')
#####################################################################
sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv'

sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'

sInputFileName3='01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv'
#####################################################################
sOutputFileName='Assess-Network-Routing-Company.csv'

Company='01-Vermeulen'
#####################################################################
##################################################################### ###

Import Country Data
##################################################################### sFileName=

Base + '/' + Company + '/' + sInputFileName1

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

print('Loading :',sFileName)

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

CountryData=pd.read_csv(sFileName,header=0,

low_memory=False, encoding="latin-1")

print('Loaded Country:',CountryData.columns.values)

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

## Assess Country Data
#####################################################################
print('#################################')
```

```
print('Changed :',CountryData.columns.values)

CountryData.rename(columns={'Country': 'Country_Name'}, inplace=True)
CountryData.rename(columns={'ISO-2-CODE': 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inplace=True)

CountryData.drop('ISO-3-Code', axis=1, inplace=True)

CountryData.drop('RowID', axis=1, inplace=True)

print('To :',CountryData.columns.values)

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

### Import Company Data
#######################################################################

sFileName=Base + '/' + Company + '/' + sInputFileName2
print('#################################')

print('Loading :',sFileName)

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

## Assess Company Data
#######################################################################
print('#################################')

print('Changed :',CompanyData.columns.values)

CompanyData.rename(columns={'Country': 'Country_Code'}, inplace=True)

print('To :',CompanyData.columns.values)

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

Import Customer Data
#######################################################################

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


sInputFileName3 print('#################################')

print('Loading :',sFileName)

print('#################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1") print('#################################')
```

```
print('Loaded Customer :',CustomerRawData.columns.values)
print('###################################')
#####################################################################
CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('###################################')

print('Remove Blank Country Code')

print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])
print('###################################')
#####################################################################
print('###################################')

print('Changed :',CustomerData.columns.values)

CustomerData.rename(columns={'Country': 'Country_Code'}, inplace=True)

print('To :',CustomerData.columns.values)

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

print('Merge Company and Country Data')

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

CompanyNetworkData=pd.merge(CompanyData, CountryData, how='inner',
on='Country_Code' )
#####################################################################
print('###################################')

print('Change ',CompanyNetworkData.columns.values)

for i in

CompanyNetworkData.columns.values: j='Company_'+i

CompanyNetworkData.rename(columns={i: j}, inplace=True)

print('To ', CompanyNetworkData.columns.values)

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not
os.path.exists(sFileDir): os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName

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

print('Storing :', sFileName) print('###################################')
CompanyNetworkData.to_csv(sFileName, index = False, encoding="latin-1")
```

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

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

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

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

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

```
###################### Assess-Network-Routing-Customer.py ######################

import sys import os import pandas as pd
#####################################################################
pd.options.mode.chained_assignment = None
#####################################################################

Base='C:/VKHCG' print('####################################')

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

print('####################################')
#####################################################################
sInputFileName=Base+'/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-
RoutingCustomer.csv'
#####################################################################
sOutputFileName='Assess-Network-Routing-Customer.gml'

Company='01-Vermeulen'
##################################################################### ###

Import Country Data
#####################################################################
sFileName=sInputFileName

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

print('Loading :',sFileName)
```

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

print(CustomerData.head())

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

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

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

Output

Assess-Network-Routing-Customer.csv

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

Assess-Network-Routing-Node.py

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

pd.options.mode.chained_assignment = None
#########################################################################

Base='C:/VKHCG'
#########################################################################
print('#################################')

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

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

sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
#########################################################################
sOutputFileName='Assess-Network-Routing-Node.csv' Company='01-Vermeulen'
######################################################################### ###
```

```
Import IP Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('###################################')

print('Loading :',sFileName)

print('###################################')
IPData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")

print('Loaded IP :', IPData.columns.values)

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

print('Changed :',IPData.columns.values)

IPData.drop('RowID', axis=1, inplace=True)

IPData.drop('ID', axis=1, inplace=True)

IPData.rename(columns={'Country': 'Country_Code'}, inplace=True)
IPData.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
IPData.rename(columns={'Post.Code': 'Post_Code'}, inplace=True)
IPData.rename(columns={'First.IP.Number': 'First_IP_Number'}, inplace=True)
IPData.rename(columns={'Last.IP.Number': 'Last_IP_Number'}, inplace=True)

print('To :',IPData.columns.values)

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

print('Change ',IPData.columns.values)

for i in

IPData.columns.values: j='Node_'+i IPData.rename(columns={i: j}, inplace=True)

print('To ', IPData.columns.values)

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not
os.path.exists(sFileDir): os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName

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

print('Storing :', sFileName) print('###################################')
```

```
IPData.to_csv(sFileName, index = False, encoding="latin-1")
#########################################################################
print('####################################')

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

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

Output:

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

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

5D. Write a Python / R program to pick the content for Bill Boards from the given data.

Code:

```
################## Assess-DE-Billboard.py ######################

import sys import os

import sqlite3 as sq

import pandas as pd

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

Base='C:/VKHCG'

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

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

print('##################################')
#####################################################################
sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
sOutputFileName='Assess-DE-Billboard-Visitor.csv' Company='02-Krennwallner'
#####################################################################

sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'

conn = sq.connect(sDatabaseName)
##################################################################### ###

Import Billboard Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('##################################')

print('Loading :',sFileName)

print('##################################')
BillboardRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1") BillboardRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BillboardData=BillboardRawData
```

```
print('Loaded Company :',BillboardData.columns.values)
print('###################################')
#####################################################################
print('#################')

sTable='Assess_BillboardData'

print('Storing :',sDatabaseName,' Table:',sTable)

BillboardData.to_sql(sTable, conn, if_exists="replace")

print('#################')
#####################################################################
print(BillboardData.head())

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

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

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

Import Billboard Data
#####################################################################
sFileName=Base + '/' + Company + '/' +

sInputFileName2 print('###################################')

print('Loading :',sFileName)

print('###################################')
VisitorRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData[VisitorRawData.Country=='DE']

print('Loaded Company :',VisitorData.columns.values)

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

sTable='Assess_VisitorData'

print('Storing :',sDatabaseName,' Table:',sTable)

VisitorData.to_sql(sTable, conn, if_exists="replace")

print('#################')
#####################################################################
print(VisitorData.head())

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

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

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

sTable='Assess_BillboardVisitorData'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="select distinct" sSQL=sSQL+ " A.Country AS BillboardCountry,"

sSQL=sSQL+ " A.Place_Name AS BillboardPlaceName,"

sSQL=sSQL+ " A.Latitude AS BillboardLatitude, "

sSQL=sSQL+ " A.Longitude AS BillboardLongitude,"

sSQL=sSQL+ " B.Country AS VisitorCountry,"

sSQL=sSQL+ " B.Place_Name AS VisitorPlaceName,"

sSQL=sSQL+ " B.Latitude AS VisitorLatitude, "

sSQL=sSQL+ " B.Longitude AS VisitorLongitude,"

sSQL=sSQL+ " (B.Last_IP_Number - B.First_IP_Number) * 365.25 * 24 * 12 AS
VisitorYearRate" sSQL=sSQL+ " from"

sSQL=sSQL+ " Assess_BillboardData as A"

sSQL=sSQL+ " JOIN "

sSQL=sSQL+ " Assess_VisitorData as B"

sSQL=sSQL+ " ON "

sSQL=sSQL+ " A.Country = B.Country"

sSQL=sSQL+ " AND "

sSQL=sSQL+ " A.Place_Name = B.Place_Name;"

BillboardVistorsData=pd.read_sql_query(sSQL, conn)

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

sTable='Assess_BillboardVistorsData'

print('Storing :',sDatabaseName,' Table:',sTable)

BillboardVistorsData.to_sql(sTable, conn, if_exists="replace")

print('#################')
#####################################################################
print(BillboardVistorsData.head())

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

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

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not
os.path.exists(sFileDir): os.makedirs(sFileDir)
#########################################################################
print('#####################################')

print('Storing :', sFileName)

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

sFileName=sFileDir + '/' + sOutputFileName BillboardVistorsData.to_csv(sFileName, index
= False) print('#####################################')
#########################################################################

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

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

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

Retrieve_Online_Visitor

SQLite Visitor's Database
C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db Table:

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

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

Assess-DE-Billboard-Visitor

E. Write a Python / R program to generate GML file from the given csv file. Understanding Your Online Visitor Data

```
import sys

import os

import sqlite3 as sq

import pandas as pd from geopy.distance

import vincenty
#####################################################################
Base='C:/VKHCG'

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

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

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

Company='02-Krennwallner'

sTable='Assess_BillboardVisitorData'

sOutputFileName='Assess-DE-Billboard-Visitor.gml'
#####################################################################

sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not
os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db' conn = sq.connect(sDatabaseName)
#####################################################################
print('#################')

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="select " sSQL=sSQL+ " A.BillboardCountry,"

sSQL=sSQL+ " A.BillboardPlaceName,"

sSQL=sSQL+ " ROUND(A.BillboardLatitude,3) AS BillboardLatitude, "

sSQL=sSQL+ " ROUND(A.BillboardLongitude,3) AS BillboardLongitude,"

sSQL=sSQL+ " (CASE WHEN A.BillboardLatitude < 0 THEN "

sSQL=sSQL+ " 'S' || ROUND(ABS(A.BillboardLatitude),3)"

sSQL=sSQL+ " ELSE " sSQL=sSQL+ " 'N' || ROUND(ABS(A.BillboardLatitude),3)"

sSQL=sSQL+ " END ) AS sBillboardLatitude,"

sSQL=sSQL+ " (CASE WHEN A.BillboardLongitude < 0 THEN "

sSQL=sSQL+ " 'W' || ROUND(ABS(A.BillboardLongitude),3)"
```

```
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ " 'E' || ROUND(ABS(A.BillboardLongitude),3)"
sSQL=sSQL+ " END ) AS sBillboardLongitude,"
sSQL=sSQL+ " A.VisitorCountry,"
sSQL=sSQL+ " A.VisitorPlaceName,"
sSQL=sSQL+ " ROUND(A.VisitorLatitude,3) AS VisitorLatitude, "
sSQL=sSQL+ " ROUND(A.VisitorLongitude,3) AS VisitorLongitude,"
sSQL=sSQL+ " (CASE WHEN A.VisitorLatitude < 0 THEN "
sSQL=sSQL+ " 'S' || ROUND(ABS(A.VisitorLatitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ " 'N' ||ROUND(ABS(A.VisitorLatitude),3)"
sSQL=sSQL+ " END ) AS sVisitorLatitude,"
sSQL=sSQL+ " (CASE WHEN A.VisitorLongitude < 0 THEN "
sSQL=sSQL+ " 'W' || ROUND(ABS(A.VisitorLongitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ " 'E' || ROUND(ABS(A.VisitorLongitude),3)"
sSQL=sSQL+ " END ) AS sVisitorLongitude,"
sSQL=sSQL+ " A.VisitorYearRate"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_BillboardVistorsData AS A;"
BillboardVistorsData=pd.read_sql_query(sSQL, conn)
print('#################')
#########################################################################
BillboardVistorsData['Distance']=BillboardVistorsData.apply(lambda row:
round(vincenty((row['BillboardLatitude'],row['BillboardLongitude']),
(row['VisitorLatitude'],row['VisitorLongitude'])).miles ,4) ,axis=1)
#########################################################################
G=nx.Graph()
#########################################################################
for i in
range(BillboardVistorsData.shape[0]):
sNode0='MediaHub-' + BillboardVistorsData['BillboardCountry'][i]
```

```
sNode1='B-'+ BillboardVistorsData['sBillboardLatitude'][i] + '-'

sNode1=sNode1 + BillboardVistorsData['sBillboardLongitude'][i]

G.add_node(sNode1, Nodetype='Billboard',

Country=BillboardVistorsData['BillboardCountry'][i],
PlaceName=BillboardVistorsData['BillboardPlaceName'][i],
Latitude=round(BillboardVistorsData['BillboardLatitude'][i],3),
Longitude=round(BillboardVistorsData['BillboardLongitude'][i],3))

sNode2='M-'+ BillboardVistorsData['sVisitorLatitude'][i] + '-'

sNode2=sNode2 + BillboardVistorsData['sVisitorLongitude'][i]

G.add_node(sNode2, Nodetype='Mobile',

Country=BillboardVistorsData['VisitorCountry'][i],
PlaceName=BillboardVistorsData['VisitorPlaceName'][i],
Latitude=round(BillboardVistorsData['VisitorLatitude'][i],3),
Longitude=round(BillboardVistorsData['VisitorLongitude'][i],3))

print('Link Media Hub :',sNode0,' to Billboard : ', sNode1)

G.add_edge(sNode0,sNode1)

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

print("Nodes of graph: ",nx.number_of_nodes(G))

print("Edges of graph: ",nx.number_of_edges(G))

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not
os.path.exists(sFileDir): os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName

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

print('Storing :', sFileName)

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

nx.write_gml(G,sFileName)

sFileName=sFileName +'.gz' nx.write_gml(G,sFileName)
#####################################################################
#####################################################################
```

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

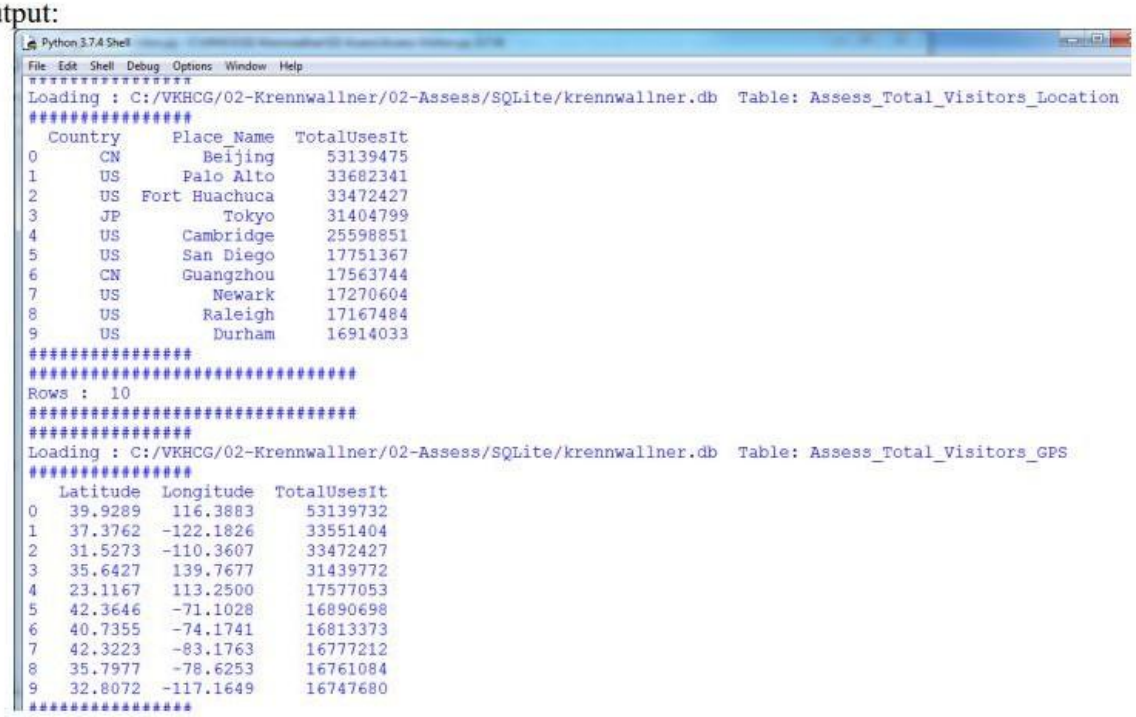
Output:

```
Assess ##############################################################################
import sys import os

import sqlite3 as sq

import pandas as pd from pandas.io

import sql ##########################################################################
Base='C:/VKHCG'

sSQL=sSQL+ " SELECT"

sSQL=sSQL+ " A.Country,"

sSQL=sSQL+ " A.Place_Name,"

sSQL=sSQL+ " A.Latitude,"

sSQL=sSQL+ " A.Longitude,"

sSQL=sSQL+ " (A.Last_IP_Number - A.First_IP_Number) AS UsesIt"

sSQL=sSQL+ " FROM"

sSQL=sSQL+ " Assess_Visitor as A"

sSQL=sSQL+ " WHERE"

sSQL=sSQL+ " Country is not null"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " Place_Name is not null;"

sql.execute(sSQL,conn)
#####################################################################
print('#################')

sView='Assess_Total_Visitors_Location'

print('Creating :',sDatabaseName,' View:',sView)

sSQL="DROP VIEW IF EXISTS " + sView + ";"

sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"

sSQL=sSQL+ " SELECT"

 sSQL=sSQL+ " Country,"

sSQL=sSQL+ " Place_Name,"
```

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

```
print('##################')
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " +
sTable + ";" TopData=pd.read_sql_query(sSQL, conn)
print('##################')
print(TopData)
print('##################')
print('###################################')
print('Rows : ',TopData.shape[0])
print('###################################')
#########################################################################
print('### Done!! ################################################')
```

Output:



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

5F. Write a Python / R program to plan the locations of the warehouses from the given data.

```
import os
import pandas as pd from geopy.geocoders
import Nominatim geolocator = Nominatim()
#####################################################################
InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_GB_Postcode_Warehouse.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_GB_Warehouse_Address.csv'
Company='03-Hillman'
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using Windows')
print('##################################')
#####################################################################
sFileDir=Base + '/' + Company + '/' +
EDSDir if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sFileDir=Base + '/' + Company + '/' +
OutputDir if not os.path.exists(sFileDir): os.makedirs(sFileDir)
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('############')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False)
Warehouse.sort_values(by='postcode', ascending=1)
WarehouseGoodHead=Warehouse[Warehouse.latitude != 0].head(5)
WarehouseGoodTail=Warehouse[Warehouse.latitude != 0].tail(5)
#####################################################################
WarehouseGoodHead['Warehouse_Point']=WarehouseGoodHead.apply(lambda row:
```

```
(str(row['latitude'])+','+str(row['longitude'])) ,axis=1)
WarehouseGoodHead['Warehouse_Address']=WarehouseGoodHead.apply(lambda row:
geolocator.reverse(row['Warehouse_Point']).address,axis=1)
WarehouseGoodHead.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodHead.drop('id', axis=1, inplace=True)

WarehouseGoodHead.drop('postcode', axis=1, inplace=True)
#########################################################################
WarehouseGoodTail['Warehouse_Point']=WarehouseGoodTail.apply(lambda
row:(str(row['latitude'])+','+str(row['longitude'])) ,axis=1)
WarehouseGoodTail['Warehouse_Address']=WarehouseGoodTail.apply(lambda
row:geolocator.reverse(row['Warehouse_Point']).address ,axis=1)
WarehouseGoodTail.drop('Warehouse_Point', axis=1, inplace=True)

WarehouseGoodTail.drop('id', axis=1, inplace=True)

WarehouseGoodTail.drop('postcode', axis=1, inplace=True)
#########################################################################
WarehouseGood=WarehouseGoodHead.append(WarehouseGoodTail, ignore_index=True)
print(WarehouseGood)
#########################################################################

sFileName=sFileDir + '/' +

OutputFileName

WarehouseGood.to_csv(sFileName, index = False)
#########################################################################

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

Output:

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

5G. Write a Python / R program using data science via clustering to determine new warehouses using the given data

```
import sys

import os

import pandas as pd ##################################################################
Base='C:/VKHCG' ##################################################################
print('##################################')

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

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


Company='03-Hillman' InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_All_Countries.csv'

EDSDir='02-Assess/01-EDS'

OutputDir=EDSDir + '/02-Python'

OutputFileName='Assess_All_Warehouse.csv'
##################################################################

sFileDir=Base + '/' + Company + '/' +

EDSDir if not os.path.exists(sFileDir): os.makedirs(sFileDir)
##################################################################

sFileDir=Base + '/' + Company + '/' +

OutputDir if not os.path.exists(sFileDir): os.makedirs(sFileDir)
##################################################################

sFileName=Base + '/' + Company + '/' + InputDir + '/' +

InputFileName

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

print('Loading :',sFileName)

Warehouse=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
##################################################################

sColumns={'X1' : 'Country',

'X2' : 'PostCode',

'X3' : 'PlaceName',

'X4' : 'AreaName',

'X5' : 'AreaCode',
```

'X10' : 'Latitude'

'X11' : 'Longitude'}

Warehouse.rename(columns=sColumns,inplace=True)

WarehouseGood=Warehouse
##

sFileName=sFileDir + '/' +

OutputFileName WarehouseGood.to_csv(sFileName, index = False)
##

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

Output:

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

Open Assess0_All_Warehose.csv from C:\VKHCG\03-Hillman\02-Assess\01-EDS\02-Pytho

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

Assess_All_Warehouse

5H. Using the given data, write a Python / R program to plan the shipping routes for best-fit international logistics.

```
import sys
import os
import pandas as pd
import networkx as nx from geopy.distance
import vincenty
import sqlite3 as sq from pandas.io
import sql ###########################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG' else:
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
############################################################################
Company='03-Hillman'
InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_All_Countries.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_Best_Logistics.gml'
############################################################################
sFileDir=Base + '/' + Company + '/' +
EDSDir if not os.path.exists(sFileDir): os.makedirs(sFileDir)
############################################################################
sFileDir=Base + '/' + Company + '/' +
OutputDir if not os.path.exists(sFileDir): os.makedirs(sFileDir)
############################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
```

```
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'

conn = sq.connect(sDatabaseName)

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

InputDir + '/' + InputFileName

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

print('Loading :',sFileName)

Warehouse=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#####################################################################

sColumns={'X1' : 'Country',

'X2' : 'PostCode',

'X3' : 'PlaceName',

'X4' : 'AreaName',

'X5' : 'AreaCode',

'X10' : 'Latitude',

'X11' : 'Longitude'}

Warehouse.rename(columns=sColumns,inplace=True)

WarehouseGood=Warehouse

#print(WarehouseGood.head())
#####################################################################
RoutePointsCountry=pd.DataFrame(WarehouseGood.groupby(['Country'])[['Latitude','Longit
ude' ]].mean())

#print(RoutePointsCountry.head())

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

sTable='Assess_RoutePointsCountry'

print('Storing :',sDatabaseName,' Table:',sTable)

RoutePointsCountry.to_sql(sTable, conn, if_exists="replace")

print('#################')
#####################################################################
RoutePointsPostCode=pd.DataFrame(WarehouseGood.groupby(['Country',
'PostCode'])[['Latitude','Longitude']].mean())

#print(RoutePointsPostCode.head())

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

```
sTable='Assess_RoutePointsPostCode'
print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsPostCode.to_sql(sTable, conn, if_exists="replace")
print('#################')
#########################################################################
RoutePointsPlaceName=pd.DataFrame(WarehouseGood.groupby(['Country',
'PostCode','PlaceName'])[['Latitude','Longitude']].mean())
#print(RoutePointsPlaceName.head())
print('#################')
sTable='Assess_RoutePointsPlaceName'
print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsPlaceName.to_sql(sTable, conn, if_exists="replace")
print('#################')
#########################################################################
### Fit Country to Country
#########################################################################
print('#################')
sView='Assess_RouteCountries'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " S.Country AS SourceCountry,"
sSQL=sSQL+ " S.Latitude AS SourceLatitude,"
sSQL=sSQL+ " S.Longitude AS SourceLongitude,"
sSQL=sSQL+ " T.Country AS TargetCountry,"
sSQL=sSQL+ " T.Latitude AS TargetLatitude,"
sSQL=sSQL+ " T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_RoutePointsCountry AS S"
sSQL=sSQL+ " ,"
```

```
sSQL=sSQL+ " Assess_RoutePointsCountry AS T"


sSQL=sSQL+ " WHERE S.Country <> T.Country"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)
('#################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
RouteCountries=pd.read_sql_query(sSQL, conn)
RouteCountries['Distance']=RouteCountries.apply(lambda row:
round(vincenty((row['SourceLatitude'],row['SourceLongitude']),
(row['TargetLatitude'],row['TargetLongitude'])).miles,4),axis=1)
print(RouteCountries.head(5))
#########################################################################
### Fit Country to Post Code
#########################################################################
print('#################')
sView='Assess_RoutePostCode'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " S.Country AS SourceCountry,"
sSQL=sSQL+ " S.Latitude AS SourceLatitude,"
sSQL=sSQL+ " S.Longitude AS SourceLongitude,
```

```
sSQL=sSQL+ " T.Country AS TargetCountry,"
sSQL=sSQL+ " T.PostCode AS TargetPostCode,"
sSQL=sSQL+ " T.Latitude AS TargetLatitude,"
sSQL=sSQL+ " T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_RoutePointsCountry AS S"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_RoutePointsPostCode AS T"
sSQL=sSQL+ " WHERE S.Country = T.Country"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)
print('#################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
RoutePostCode=pd.read_sql_query(sSQL, conn)
RoutePostCode['Distance']=RoutePostCode.apply(lambda row:
round(vincenty((row['SourceLatitude'],row['SourceLongitude']),
(row['TargetLatitude'],row['TargetLongitude'])).miles ,4) ,axis=1)
print(RoutePostCode.head(5))
#########################################################################
### Fit Post Code to Place Name
#########################################################################
print('#################')
sView='Assess_RoutePlaceName'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
```

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

```
sSQL=sSQL+ " " + sView + ";"

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

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

G=nx.Graph()
#########################################################################
print('Countries:',RouteCountries.shape)

for i in range(RouteCountries.shape[0]):

sNode0='C-' + RouteCountries['SourceCountry'][i]

G.add_node(sNode0, Nodetype='Country',

Country=RouteCountries['SourceCountry'][i],

Latitude=round(RouteCountries['SourceLatitude'][i],4),
Longitude=round(RouteCountries['SourceLongitude'][i],4))

sNode1='C-' + RouteCountries['TargetCountry'][i]

G.add_node(sNode1, Nodetype='Country',

Country=RouteCountries['TargetCountry'][i],

Latitude=round(RouteCountries['TargetLatitude'][i],4),
Longitude=round(RouteCountries['TargetLongitude'][i],4))
G.add_edge(sNode0,sNode1,distance=round(RouteCountries['Distance'][i],3))
#print(sNode0,sNode1)
######################################################################### print('Post
Code:',RoutePostCode.shape)

for i in range(RoutePostCode.shape[0]):

sNode0='C-' + RoutePostCode['SourceCountry'][i]

G.add_node(sNode0, Nodetype='Country',

Country=RoutePostCode['SourceCountry'][i],

Latitude=round(RoutePostCode['SourceLatitude'][i],4),
Longitude=round(RoutePostCode['SourceLongitude'][i],4))

sNode1='P-' + RoutePostCode['TargetPostCode'][i] + '-' + RoutePostCode['TargetCountry'][i]
G.add_node(sNode1, Nodetype='PostCode', Country=RoutePostCode['TargetCountry'][i],
PostCode=RoutePostCode['TargetPostCode'][i],
Latitude=round(RoutePostCode['TargetLatitude'][i],4),
Longitude=round(RoutePostCode['TargetLongitude'][i],4))
G.add_edge(sNode0,sNode1,distance=round(RoutePostCode['Distance'][i],3))
```

```
#print(sNode0,sNode1)
###################################################################### print('Place
Name:',RoutePlaceName.shape)

for i in range(RoutePlaceName.shape[0]):

sNode0='P-' + RoutePlaceName['TargetPostCode'][i] + '-' sNode0=sNode0 +
RoutePlaceName['TargetCountry'][i]

G.add_node(sNode0, Nodetype='PostCode',

Country=RoutePlaceName['SourceCountry'][i],

PostCode=RoutePlaceName['TargetPostCode'][i],
Latitude=round(RoutePlaceName['SourceLatitude'][i],4),
Longitude=round(RoutePlaceName['SourceLongitude'][i],4))

sNode1='L-' + RoutePlaceName['TargetPlaceName'][i] + '-' sNode1=sNode1 +
RoutePlaceName['TargetPostCode'][i] + '-'

sNode1=sNode1 + RoutePlaceName['TargetCountry'][i]

G.add_node(sNode1, Nodetype='PlaceName',

Country=RoutePlaceName['TargetCountry'][i],

PostCode=RoutePlaceName['TargetPostCode'][i],
PlaceName=RoutePlaceName['TargetPlaceName'][i],
Latitude=round(RoutePlaceName['TargetLatitude'][i],4),
Longitude=round(RoutePlaceName['TargetLongitude'][i],4))

G.add_edge(sNode0,sNode1,distance=round(RoutePlaceName['Distance'][i],3))
#print(sNode0,sNode1)
######################################################################

sFileName=sFileDir + '/' +

OutputFileName

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

print('Storing :', sFileName)

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

nx.write_gml(G,sFileName)

sFileName=sFileName +'.gz' nx.write_gml(G,sFileName)
######################################################################
print('##################################')

print('Path:', nx.shortest_path(G,source='P-SW1-GB',target='P-01001-US',weight='distance'))
print('Path length:', nx.shortest_path_length(G,source='P-SW1-GB',target='P-01001-
US',weight='distance'))
```

```
print('Path length (1):', nx.shortest_path_length(G,source='P-SW1-
GB',target='CGB',weight='distance'))

print('Path length (2):', nx.shortest_path_length(G,source='C-
GB',target='CUS',weight='distance')) print('Path length (3):',
nx.shortest_path_length(G,source='C-US',target='P-01001- US',weight='distance'))

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

print('Routes from P-SW1-GB < 2: ', nx.single_source_shortest_path(G,source='P-SW1-GB'
,cutoff=1)) print('Routes from P-01001-US < 2: ',
nx.single_source_shortest_path(G,source='P-01001-US' ,cutoff=1))

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

print('Vacuum Database')

sSQL="VACUUM;"

sql.execute(sSQL,conn)

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

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

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

5I) Write a Python / R program to decide the best packing option to ship in container from the given data

```
import sys

import os

import pandas as pd

import sqlite3 as sq from pandas.io

import sql ########################################################################
Base='C:/VKHCG' print('#################################')

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

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

Company='03-Hillman'

InputDir='01-Retrieve/01-EDS/02-Python'

InputFileName1='Retrieve_Product.csv'

InputFileName2='Retrieve_Box.csv'

InputFileName3='Retrieve_Container.csv'

EDSDir='02-Assess/01-EDS'

OutputDir=EDSDir + '/02-Python'

OutputFileName='Assess_Shipping_Containers.csv'
########################################################################

sFileDir=Base + '/' + Company + '/' +

EDSDir if not os.path.exists(sFileDir): os.makedirs(sFileDir)
########################################################################

sFileDir=Base + '/' + Company + '/' +

if not os.path.exists(sFileDir): os.makedirs(sFileDir)
########################################################################

sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not
os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
########################################################################
sDatabaseName=sDataBaseDir + '/hillman.db'

conn = sq.connect(sDatabaseName)

Import Product Data
########################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName1
```

```
print('############') print('Loading :',sFileName) ProductRawData=pd.read_csv(sFileName,
header=0, low_memory=False, encoding="latin-1" )

ProductRawData.drop_duplicates(subset=None, keep='first', inplace=True)

ProductRawData.index.name = 'IDNumber'

ProductData=ProductRawData[ProductRawData.Length <= 0.5].head(10)

print('Loaded Product :',ProductData.columns.values)

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

sTable='Assess_Product'

print('Storing :',sDatabaseName,' Table:',sTable)

ProductData.to_sql(sTable, conn, if_exists="replace")

print('#################')
#########################################################################
print(ProductData.head())

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

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

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

Import Box Data
#########################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName2

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

print('Loading :',sFileName)

BoxRawData=pd.read_csv(sFileName, header=0, low_memory=False, encoding="latin-1" )
BoxRawData.drop_duplicates(subset=None, keep='first', inplace=True)

BoxRawData.index.name = 'IDNumber'

BoxData=BoxRawData[BoxRawData.Length <= 1].head(1000)

print('Loaded Product :',BoxData.columns.values)

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

sTable='Assess_Box'

print('Storing :',sDatabaseName,' Table:',sTable)
```

```
BoxData.to_sql(sTable, conn, if_exists="replace")


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

print(BoxData.head())

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

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

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

Import Container Data
#########################################################################

sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName3

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

print('Loading :',sFileName)

ContainerRawData=pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1" ) ContainerRawData.drop_duplicates(subset=None, keep='first',
inplace=True) ContainerRawData.index.name = 'IDNumber'
ContainerData=ContainerRawData[ContainerRawData.Length <= 2].head(10)

print('Loaded Product :',ContainerData.columns.values)
print('###################################')
#########################################################################
print('#################')

sTable='Assess_Container'

print('Storing :',sDatabaseName,' Table:',sTable)

BoxData.to_sql(sTable, conn, if_exists="replace")

print('#################')
#########################################################################
print(ContainerData.head())

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

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

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

```
### Fit Product in Box
#########################################################################
print('#################')

sView='Assess_Product_in_Box'

print('Creating :',sDatabaseName,' View:',sView)

sSQL="DROP VIEW IF EXISTS " + sView + ";"

sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"

sSQL=sSQL+ " SELECT"

sSQL=sSQL+ " P.UnitNumber AS ProductNumber,"

sSQL=sSQL+ " B.UnitNumber AS BoxNumber,"

sSQL=sSQL+ " (B.Thickness * 1000) AS PackSafeCode,"

sSQL=sSQL+ " (B.BoxVolume - P.ProductVolume) AS PackFoamVolume,"

sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 167 AS
Air_Dimensional_Weight," sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) *
(B.Height*10)) * 333 AS Road_Dimensional_Weight," sSQL=sSQL+ " ((B.Length*10) *
(B.Width*10) * (B.Height*10)) * 1000 AS Sea_Dimensional_Weight," sSQL=sSQL+ "
P.Length AS Product_Length,"

sSQL=sSQL+ " P.Width AS Product_Width,"

sSQL=sSQL+ " P.Height AS Product_Height,"

sSQL=sSQL+ " P.ProductVolume AS Product_cm_Volume,"

sSQL=sSQL+ " ((P.Length*10) * (P.Width*10) * (P.Height*10)) AS
Product_ccm_Volume," sSQL=sSQL+ " (B.Thickness * 0.95) AS Minimum_Pack_Foam,"

sSQL=sSQL+ " (B.Thickness * 1.05) AS Maximum_Pack_Foam,"

sSQL=sSQL+ " B.Length - (B.Thickness * 1.10) AS Minimum_Product_Box_Length,"

sSQL=sSQL+ " B.Length - (B.Thickness * 0.95) AS Maximum_Product_Box_Length,"

sSQL=sSQL+ " B.Width - (B.Thickness * 1.10) AS Minimum_Product_Box_Width,"

sSQL=sSQL+ " B.Width - (B.Thickness * 0.95) AS Maximum_Product_Box_Width,"

sSQL=sSQL+ " B.Height - (B.Thickness * 1.10) AS Minimum_Product_Box_Height,"

sSQL=sSQL+ " B.Height - (B.Thickness * 0.95) AS Maximum_Product_Box_Height,"

sSQL=sSQL+ " B.Length AS Box_Length,"

sSQL=sSQL+ " B.Width AS Box_Width,"

sSQL=sSQL+ " B.Height AS Box_Height,"
```

```
sSQL=sSQL+ " B.BoxVolume AS Box_cm_Volume,"

sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) AS Box_ccm_Volume,"

sSQL=sSQL+ " (2 * B.Length * B.Width) + (2 * B.Length * B.Height) + (2 * B.Width *
B.Height) AS Box_sqm_Area,"

sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 3.5 AS
Box_A_Max_Kg_Weight," sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) *
(B.Height*10)) * 7.7 AS Box_B_Max_Kg_Weight," sSQL=sSQL+ " ((B.Length*10) *
(B.Width*10) * (B.Height*10)) * 10.0 AS Box_C_Max_Kg_Weight" sSQL=sSQL+ "
FROM"

sSQL=sSQL+ " Assess_Product as P"

sSQL=sSQL+ " ,"

sSQL=sSQL+ " Assess_Box as B"

sSQL=sSQL+ " WHERE"

sSQL=sSQL+ " P.Length >= (B.Length - (B.Thickness * 1.10))"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " P.Width >= (B.Width - (B.Thickness * 1.10))"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " P.Height >= (B.Height - (B.Thickness * 1.10))"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " P.Length <= (B.Length - (B.Thickness * 0.95))"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " P.Width <= (B.Width - (B.Thickness * 0.95))"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " P.Height <= (B.Height - (B.Thickness * 0.95))"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " (B.Height - B.Thickness) >= 0"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " (B.Width - B.Thickness) >= 0"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " (B.Height - B.Thickness) >= 0"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " B.BoxVolume >= P.ProductVolume;"
```

```
sql.execute(sSQL,conn)
#########################################################################

### Fit Box in Pallet
######################################################################### t=0 for l in
range(2,8):

for w in range(2,8):

for h in range(4): t += 1 PalletLine=[('IDNumber',[t]), ('ShipType', ['Pallet']), ('UnitNumber',
('L-'+format(t,"06d"))), ('Box_per_Length',(format(2**l,"4d"))),
('Box_per_Width',(format(2**w,"4d"))), ('Box_per_Height',(format(2**h,"4d")))]

if t==1: PalletFrame = pd.DataFrame.from_items(PalletLine)

else: PalletRow = pd.DataFrame.from_items(PalletLine) PalletFrame =
PalletFrame.append(PalletRow)

PalletFrame.set_index(['IDNumber'],inplace=True)
#########################################################################

PalletFrame.head()

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

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

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

### Fit Box on Pallet
#########################################################################
print('#################')

sView='Assess_Box_on_Pallet'

print('Creating :',sDatabaseName,' View:',sView)

sSQL="DROP VIEW IF EXISTS " + sView + ";"

sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"

sSQL=sSQL+ " SELECT DISTINCT"

sSQL=sSQL+ " P.UnitNumber AS PalletNumber,"

sSQL=sSQL+ " B.UnitNumber AS BoxNumber,"

sSQL=sSQL+ " round(B.Length*P.Box_per_Length,3) AS Pallet_Length,"

sSQL=sSQL+ " round(B.Width*P.Box_per_Width,3) AS Pallet_Width,"

sSQL=sSQL+ " round(B.Height*P.Box_per_Height,3) AS Pallet_Height,"
```

```
sSQL=sSQL+ " P.Box_per_Length * P.Box_per_Width * P.Box_per_Height AS
Pallet_Boxes" sSQL=sSQL+ " FROM"

sSQL=sSQL+ " Assess_Box as B"

sSQL=sSQL+ " ,"

sSQL=sSQL+ " Assess_Pallet as P"

sSQL=sSQL+ " WHERE"

sSQL=sSQL+ " round(B.Length*P.Box_per_Length,3) <= 20"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " round(B.Width*P.Box_per_Width,3) <= 9"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " round(B.Height*P.Box_per_Height,3) <= 5;"

sql.execute(sSQL,conn)
########################################################################
sTables=['Assess_Product_in_Box','Assess_Box_on_Pallet']

for sTable in sTables:

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

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL=" SELECT "

sSQL=sSQL+ " *"

sSQL=sSQL+ " FROM"

sSQL=sSQL+ " " + sTable + ";"

SnapShotData=pd.read_sql_query(sSQL, conn)

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

sTableOut=sTable + '_SnapShot'

print('Storing :',sDatabaseName,' Table:',sTable)

SnapShotData.to_sql(sTableOut, conn, if_exists="replace")

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

### Fit Pallet in Container
########################################################################
sTables=['Length','Width','Height']

for sTable in sTables: sView='Assess_Pallet_in_Container_' + sTable
```

```
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " C.UnitNumber AS ContainerNumber,"
sSQL=sSQL+ " P.PalletNumber,"
sSQL=sSQL+ " P.BoxNumber,"
sSQL=sSQL+ " round(C." + sTable + "/P.Pallet_" + sTable + ",0)"
sSQL=sSQL+ " AS Pallet_per_" + sTable + ","
sSQL=sSQL+ " round(C." + sTable + "/P.Pallet_" + sTable + ",0)"
sSQL=sSQL+ " * P.Pallet_Boxes AS Pallet_" + sTable + "_Boxes,"
sSQL=sSQL+ " P.Pallet_Boxes" sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Container as C"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Box_on_Pallet_SnapShot as P"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " round(C.Length/P.Pallet_Length,0) > 0"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " round(C.Width/P.Pallet_Width,0) > 0"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " round(C.Height/P.Pallet_Height,0) > 0;"
sql.execute(sSQL,conn)
print('#################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
SnapShotData=pd.read_sql_query(sSQL, conn)
```

```
print('#################') sTableOut= sView + '_SnapShot'

print('Storing :',sDatabaseName,' Table:',sTableOut)

SnapShotData.to_sql(sTableOut, conn, if_exists="replace")

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

sView='Assess_Pallet_in_Container'

print('Creating :',sDatabaseName,' View:',sView)

sSQL="DROP VIEW IF EXISTS " + sView + ";"

sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"

sSQL=sSQL+ " SELECT"

sSQL=sSQL+ " CL.ContainerNumber,"

sSQL=sSQL+ " CL.PalletNumber,"

sSQL=sSQL+ " CL.BoxNumber,"

sSQL=sSQL+ " CL.Pallet_Boxes AS Boxes_per_Pallet,"

sSQL=sSQL+ " CL.Pallet_per_Length,"

sSQL=sSQL+ " CW.Pallet_per_Width,"

sSQL=sSQL+ " CH.Pallet_per_Height,"

sSQL=sSQL+ " CL.Pallet_Length_Boxes * CW.Pallet_Width_Boxes *
CH.Pallet_Height_Boxes AS Container_Boxes"

sSQL=sSQL+ " FROM"

sSQL=sSQL+ " Assess_Pallet_in_Container_Length_SnapShot as CL"

sSQL=sSQL+ " JOIN"

sSQL=sSQL+ " Assess_Pallet_in_Container_Width_SnapShot as CW"

sSQL=sSQL+ " ON"

sSQL=sSQL+ " CL.ContainerNumber = CW.ContainerNumber"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " CL.PalletNumber = CW.PalletNumber"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " CL.BoxNumber = CW.BoxNumber"
```

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

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

J. Write a Python program to create a delivery route using the given data.

```
import sys

import os

import pandas as pd

import sqlite3 as sq from pandas.io

import sql

import networkx as nx from geopy.distance

import vincenty
########################################################################## nMax=3
nMaxPath=10 nSet=False nVSet=False
##########################################################################

Base='C:/VKHCG'
##########################################################################
print('####################################')

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

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

Company='03-Hillman' InputDir1='01-Retrieve/01-EDS/01-R'

InputDir2='01-Retrieve/01-EDS/02-Python'
InputFileName1='Retrieve_GB_Postcode_Warehouse.csv'
InputFileName2='Retrieve_GB_Postcodes_Shops.csv'

EDSDir='02-Assess/01-EDS'

OutputDir=EDSDir + '/02-Python'

OutputFileName1='Assess_Shipping_Routes.gml'

OutputFileName2='Assess_Shipping_Routes.txt'
#######################################################################

sFileDir=Base + '/' + Company + '/' +

EDSDir if not os.path.exists(sFileDir): os.makedirs(sFileDir)
#######################################################################

sFileDir=Base + '/' + Company + '/' +

OutputDir if not os.path.exists(sFileDir): os.makedirs(sFileDir)
#######################################################################

sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not
os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)
#######################################################################
sDatabaseName=sDataBaseDir + '/hillman.db'

conn = sq.connect(sDatabaseName)
#######################################################################
######################################################################## ###

Import Warehouse Data
#######################################################################

sFileName=Base + '/' + Company + '/' + InputDir1 + '/' +

InputFileName1

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

print('Loading :',sFileName)

WarehouseRawData=pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1" ) WarehouseRawData.drop_duplicates(subset=None, keep='first',
inplace=True) WarehouseRawData.index.name = 'IDNumber'

WarehouseData=WarehouseRawData.head(nMax)
WarehouseData=WarehouseData.append(WarehouseRawData.tail(nMax))
```

```
WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode
=='KA13'])

if nSet==True:
WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode
=='SW1W'])

WarehouseData.drop_duplicates(subset=None, keep='first', inplace=True) print('Loaded
Warehouses :',WarehouseData.columns.values)

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

sTable='Assess_Warehouse_UK'

print('Storing :',sDatabaseName,' Table:',sTable)

WarehouseData.to_sql(sTable, conn, if_exists="replace")

print('#################')
#####################################################################
print(WarehouseData.head())

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

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

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

Import Shop Data
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir1 + '/' +

InputFileName2

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

print('Loading :',sFileName)

ShopRawData=pd.read_csv(sFileName, header=0, low_memory=False, encoding="latin-1" )
ShopRawData.drop_duplicates(subset=None, keep='first', inplace=True)

ShopRawData.index.name = 'IDNumber'

ShopData=ShopRawData

print('Loaded Shops :',ShopData.columns.values)

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

sTable='Assess_Shop_UK'
```

```
print('Storing :',sDatabaseName,' Table:',sTable)

ShopData.to_sql(sTable, conn, if_exists="replace")

print('#################')
#########################################################################
print(ShopData.head())

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

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

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

### Connect HQ
#########################################################################
print('#################')

sView='Assess_HQ'

print('Creating :',sDatabaseName,' View:',sView)

sSQL="DROP VIEW IF EXISTS " + sView + ";"

sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"

sSQL=sSQL+ " SELECT"

sSQL=sSQL+ " W.postcode AS HQ_PostCode,"

sSQL=sSQL+ " 'HQ-' || W.postcode AS HQ_Name,"

sSQL=sSQL+ " round(W.latitude,6) AS HQ_Latitude,"

sSQL=sSQL+ " round(W.longitude,6) AS HQ_Longitude"

sSQL=sSQL+ " FROM"

sSQL=sSQL+ " Assess_Warehouse_UK as W"

sSQL=sSQL+ " WHERE"

sSQL=sSQL+ " TRIM(W.postcode) in ('KA13','SW1W');"

sql.execute(sSQL,conn)
#########################################################################

### Connect Warehouses
#########################################################################
print('#################')

sView='Assess_Warehouse'

print('Creating :',sDatabaseName,' View:',sView)
```

```
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " W.postcode AS Warehouse_PostCode,"
sSQL=sSQL+ " 'WH-' || W.postcode AS Warehouse_Name,"
sSQL=sSQL+ " round(W.latitude,6) AS Warehouse_Latitude,"
sSQL=sSQL+ " round(W.longitude,6) AS Warehouse_Longitude"
sSQL=sSQL+ " FROM" sSQL=sSQL+ " Assess_Warehouse_UK as W;"
sql.execute(sSQL,conn)
print('##################')
sView='Assess_Shop' print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " +
sView + ";" sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " TRIM(S.postcode) AS Shop_PostCode,"
sSQL=sSQL+ " 'SP-' || TRIM(S.FirstCode) || '-' || TRIM(S.SecondCode) AS Shop_Name,"
sSQL=sSQL+ " TRIM(S.FirstCode) AS Warehouse_PostCode,"
sSQL=sSQL+ " round(S.latitude,6) AS Shop_Latitude,"
sSQL=sSQL+ " round(S.longitude,6) AS Shop_Longitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Warehouse_UK as W"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " Assess_Shop_UK as S"
sSQL=sSQL+ " ON"
sSQL=sSQL+ " TRIM(W.postcode) = TRIM(S.FirstCode);"
sql.execute(sSQL,conn)
##########################################################################
##########################################################################
```

```
G=nx.Graph()
#####################################################################
print('#################')
sTable = 'Assess_HQ'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData=pd.read_sql_query(sSQL, conn)
print('#################')
#####################################################################
print(RouteData.head())
print('###################################')
print('HQ Rows : ',RouteData.shape[0])
print('###################################')
#####################################################################
for i in range(RouteData.shape[0]):
sNode0=RouteData['HQ_Name'][i]
G.add_node(sNode0, Nodetype='HQ',
PostCode=RouteData['HQ_PostCode'][i],
Latitude=round(RouteData['HQ_Latitude'][i],6),
Longitude=round(RouteData['HQ_Longitude'][i],6))
#####################################################################
print('#################')
sTable = 'Assess_Warehouse'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " +
sTable + ";"
RouteData=pd.read_sql_query(sSQL, conn)
```

```
print('#################')
#####################################################################
print(RouteData.head())

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

print('Warehouse Rows : ',RouteData.shape[0])

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

for i in range(RouteData.shape[0]):

sNode0=RouteData['Warehouse_Name'][i]

G.add_node(sNode0,

Nodetype='Warehouse',

PostCode=RouteData['Warehouse_PostCode'][i],
Latitude=round(RouteData['Warehouse_Latitude'][i],6),
Longitude=round(RouteData['Warehouse_Longitude'][i],6))

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

sTable = 'Assess_Shop'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL=" SELECT DISTINCT"

sSQL=sSQL+ " *"

sSQL=sSQL+ " FROM"

sSQL=sSQL+ " " + sTable + ";" RouteData=pd.read_sql_query(sSQL, conn)

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

print(RouteData.head())

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

print('Shop Rows : ',RouteData.shape[0])

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

for i in range(RouteData.shape[0]):

sNode0=RouteData['Shop_Name'][i]

G.add_node(sNode0, Nodetype='Shop',

PostCode=RouteData['Shop_PostCode'][i],
WarehousePostCode=RouteData['Warehouse_PostCode'][i],
Latitude=round(RouteData['Shop_Latitude'][i],6),
Longitude=round(RouteData['Shop_Longitude'][i],6))
#####################################################################
```

```
## Create Edges
#####################################################################
print('###################################')

print('Loading Edges')

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

for sNode0 in nx.nodes_iter(G):

for sNode1 in nx.nodes_iter(G):

if G.node[sNode0]['Nodetype']=='HQ' and \ G.node[sNode1]['Nodetype']=='HQ' and \
sNode0 != sNode1: distancemeters=round(\ vincenty(\ (\ G.node[sNode0]['Latitude'],\
G.node[sNode0]['Longitude']\

),\ (\ G.node[sNode1]['Latitude']\ ,\ G.node[sNode1]['Longitude']\ )\ ).meters\ ,0)

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

if distancemiles >= 0.05: cost = round(150+(distancemiles * 2.5),6) vehicle='V001' else: cost
= round(2+(distancemiles * 0.10),6) vehicle='ForkLift'
G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \ DistanceMiles=distancemiles,
\ Cost=cost,Vehicle=vehicle)

if nVSet==True:

print('Edge-H-H:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\
distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

if G.node[sNode0]['Nodetype']=='HQ' and \ G.node[sNode1]['Nodetype']=='Warehouse' and \
sNode0 != sNode1: distancemeters=round(\ vincenty(\ (\ G.node[sNode0]['Latitude'],\
G.node[sNode0]['Longitude']\ ),\ (\ G.node[sNode1]['Latitude']\ ,\
G.node[sNode1]['Longitude']\ )\ ).meters\ ,0)

distancemiles=round(\ vincenty(\ (\ G.node[sNode0]['Latitude'],\
G.node[sNode0]['Longitude']\ ),\ (\ G.node[sNode1]['Latitude']\ ,\
G.node[sNode1]['Longitude']\ )\ ).miles\ ,3) if distancemiles >= 10: cost =
round(50+(distancemiles * 2),6)

vehicle='V002' else: cost = round(5+(distancemiles * 1.5),6)

vehicle='V003' if distancemiles <= 50:
G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \ DistanceMiles=distancemiles,
\ Cost=cost,Vehicle=vehicle) if nVSet==True: print('Edge-H-W:',sNode0,' to ', sNode1, \ '
Distance:',distancemeters,'meters',\ distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

if nSet==True

and \ G.node[sNode0]['Nodetype']=='Warehouse'
```

```
and \ G.node[sNode1]['Nodetype']=='Warehouse' and \ sNode0 != sNode1:
distancemeters=round(\ vincenty(\ (\ G.node[sNode0]['Latitude'],\
G.node[sNode0]['Longitude']\ ),\ (\ G.node[sNode1]['Latitude']\ ,

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

G.node[sNode0]['Longitude']\ ),\ (\ G.node[sNode1]['Latitude']\

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

if distancemiles >= 10: cost = round(50+(distancemiles * 1.10),6) vehicle='V004' else: cost =
round(5+(distancemiles * 1.05),6)

 vehicle='V005' if distancemiles <= 20:

 G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
DistanceMiles=distancemiles, \ Cost=cost,Vehicle=vehicle)

if nVSet==True: print('Edge-W-W:',sNode0,' to ', sNode1, \ '
Distance:',distancemeters,'meters',\ distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

if G.node[sNode0]['Nodetype']=='Warehouse' and \ G.node[sNode1]['Nodetype']=='Shop'
and \ G.node[sNode0]['PostCode']==G.node[sNode1]['WarehousePostCode'] and \ sNode0 !=
sNode1: distancemeters=round(\ vincenty(\ (\ G.node[sNode0]['Latitude'],\

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

(\ G.node[sNode0]['Latitude'],\

 G.node[sNode0]['Longitude']\ ),\

 (\ G.node[sNode1]['Latitude']\ ,\

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

if distancemiles >= 10:

cost = round(50+(distancemiles * 1.50),6)

vehicle='V006' else: cost = round(5+(distancemiles * 0.75),6)

vehicle='V007' if distancemiles <= 10:

G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \ DistanceMiles=distancemiles,
\ Cost=cost,Vehicle=vehicle)

if nVSet==True:

print('Edge-W-S:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\
distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

if nSet==True

and \ G.node[sNode0]['Nodetype']=='Shop' and \
```

```
G.node[sNode1]['Nodetype']=='Shop' and \
G.node[sNode0]['WarehousePostCode']==G.node[sNode1]['WarehousePostCode'] and \

sNode0 != sNode1: distancemeters=round(\ vincenty(\ (\ G.node[sNode0]['Latitude'],\
G.node[sNode0]['Longitude']\ ),\ (\ G.node[sNode1]['Latitude']\ ,\
G.node[sNode1]['Longitude']\ )\ ).meters\ ,0) distancemiles=round(\ vincenty(\ (\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\ ),\ (\ G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\ )\ ).miles\ ,3)

if distancemiles >= 0.05: cost = round(5+(distancemiles * 0.5),6)

vehicle='V008' else: cost = round(1+(distancemiles * 0.1),6)

vehicle='V009

if distancemiles <= 0.075: G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
DistanceMiles=distancemiles, \

Cost=cost,Vehicle=vehicle)

if nVSet==True: print('Edge-S-S:',sNode0,' to ', sNode1, \ '

Distance:',distancemeters,'meters',\ distancemiles,'miles','Cost',

cost,'Vehicle',vehicle)

if nSet==True and \ G.node[sNode0]['Nodetype']=='Shop' and \

G.node[sNode1]['Nodetype']=='Shop' and \
G.node[sNode0]['WarehousePostCode']!=G.node[sNode1]['WarehousePostCode'] and \

sNode0 != sNode1: distancemeters=round(\ vincenty(\ (\ G.node[sNode0]['Latitude'],\
G.node[sNode0]['Longitude']\ ),\ (\ G.node[sNode1]['Latitude']\ ,\
G.node[sNode1]['Longitude']\ )\ ).meters\ ,0) distancemiles=round(\ vincenty(\ (\
G.node[sNode0]['Latitude'],\ G.node[sNode0]['Longitude']\ ),\ (\ G.node[sNode1]['Latitude']\
,\ G.node[sNode1]['Longitude']\ )\ ).miles\ ,3)

cost = round(1+(distancemiles * 0.1),6) vehicle='V010'

if distancemiles <= 0.025:

G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \

DistanceMiles=distancemiles,

\ Cost=cost,Vehicle=vehicle)

if nVSet==True:

print('Edge-S-S:',sNode0,' to ', sNode1, \ ' Distance:',distancemeters,'meters',\

distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

sFileName=sFileDir + '/' +

OutputFileName1
```

```
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.write_gml(G,sFileName)
sFileName=sFileName +'.gz' nx.write_gml(G,sFileName)
print('Nodes:',nx.number_of_nodes(G))
print('Edges:',nx.number_of_edges(G))
sFileName=sFileDir + '/' +
OutputFileName2
print('#################################')
print('Storing :', sFileName)
print('#################################')
## Create Paths
print('#################################')
print('Loading Paths')
print('#################################') f = open(sFileName,'w') l=0
sline = 'ID|Cost|StartAt|EndAt|Path|Measure'
if nVSet==True: print ('0', sline) f.write(sline+ '\n')
for sNode0 in nx.nodes_iter(G):
for sNode1 in nx.nodes_iter(G):
if sNode0 != sNode1 and \ nx.has_path(G, sNode0, sNode1)==True and \
nx.shortest_path_length(G, \ source=sNode0, \ target=sNode1, \ weight='DistanceMiles') <
nMaxPath: l+=1 sID='{:.0f}'.format(l) spath = ','.join(nx.shortest_path(G, \ source=sNode0, \
target=sNode1, \ weight='DistanceMiles')) slength= '{:.6f}'.format(\
nx.shortest_path_length(G, \ source=sNode0, \ target=sNode1, \ weight='DistanceMiles'))
sline = sID + '|"DistanceMiles"|"' + sNode0 + '"|"' \ + sNode1 + '"|"' + spath + '"|' + slength if
nVSet==True:
print (sline) f.write(sline + '\n') l+=1 sID='{:.0f}'.format(l) spath = ','.join(nx.shortest_path(G,
\ source=sNode0, \ target=sNode1, \
weight='DistanceMeters'))
slength= '{:.6f}'.format(\
nx.shortest_path_length(G, \ source=sNode0, \
```

```
target=sNode1, \ weight='DistanceMeters'))

sline = sID + '|"DistanceMeters"|"' + sNode0 + '"|"' \ + sNode1 + '"|"' + spath + '"|' + slength

if nVSet==True:

print (sline) f.write(sline + '\n') l+=1 sID='{:.0f}'.format(l)

spath = ','.join(nx.shortest_path(G, \

source=sNode0, \

target=sNode1, \ weight='Cost'))

slength= '{:.6f}'.format(\ nx.shortest_path_length(G, \ source=sNode0, \ target=sNode1, \
weight='Cost'))

sline = sID + '|"Cost"|"' + sNode0 + '"|"' \ + sNode1 + '"|"' + spath + '"|' + slength if
nVSet==True: print (sline)

f.write(sline + '\n') f.close()

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

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

print('Paths:',sID)

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

print('Vacuum Database')

sSQL="VACUUM;"

sql.execute(sSQL,conn)

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

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

```
################################
Working Base : C:/VKHCG  using  win32
################################
###########
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_GB_Postcode_Warehouse.csv
Loaded Warehouses : ['id' 'postcode' 'latitude' 'longitude']
################################
################
Storing : C:/VKHCG/03-Hillman/02-Assess/SQLite/hillman.db  Table: Assess_Warehouse_UK
################
            id postcode  latitude  longitude
IDNumber
0            2     AB10  57.13514   -2.11731
1            3     AB11  57.13875   -2.09089
2            4     AB12  57.10100   -2.11060
3000      3003     PA80   0.00000    0.00000
3001      3004      L80   0.00000    0.00000
################################
Rows :  7
################################
###########
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_GB_Postcodes_Shops.csv
Loaded Shops : ['version https://git-lfs.github.com/spec/v1']
################################
################
Storing : C:/VKHCG/03-Hillman/02-Assess/SQLite/hillman.db  Table: Assess_Shop_UK
```

5K. Write a Python program to create Simple forex trading planner from the given data.

```
import sys

import os

import sqlite3 as sq

import pandas as pd
#####################################################################
Base='C:/VKHCG' print('###################################')

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

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

Company='04-Clark'

sInputFileName1='01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve-Country-Currency.csv'

sInputFileName2='04-Clark/01-Retrieve/01-EDS/01-R/Retrieve_Euro_EchangeRates.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'

if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/clark.db'

conn = sq.connect(sDatabaseName)
#####################################################################  ###

Import Country Data
#####################################################################
sFileName1=Base + '/' + sInputFileName1

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

print('Loading :',sFileName1) print('###################################')
CountryRawData=pd.read_csv(sFileName1,header=0,low_memory=False, encoding="latin-1") CountryRawData.drop_duplicates(subset=None, keep='first', inplace=True)
CountryData=CountryRawData

print('Loaded Company :',

CountryData.columns.values)

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

```
sTable='Assess_Country'

print('Storing :',sDatabaseName,' Table:',sTable)

CountryData.to_sql(sTable, conn,

if_exists="replace")

print('#################')
#########################################################################
print(CountryData.head())

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

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

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

Import Forex Data
########################################################################
sFileName2=Base + '/' + sInputFileName2

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

print('Loading :',sFileName2)

print('###################################')
ForexRawData=pd.read_csv(sFileName2,header=0,low_memory=False, encoding="latin-1")
ForexRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ForexData=ForexRawData.head(5)

print('Loaded Company :',ForexData.columns.values)

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

sTable='Assess_Forex'

print('Storing :',sDatabaseName,' Table:',sTable) ForexData.to_sql(sTable, conn,

if_exists="replace")

print('#################')
########################################################################
print(ForexData.head())

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

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

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

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

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

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

5L. Write a Python program to process the balance sheet to ensure that only good data is processing.

```
import sys

import os

import sqlite3 as sq

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

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

print('##################################') Company='04-Clark' sInputFileName='01-Retrieve/01-EDS/01-R/Retrieve_Profit_And_Loss.csv'
########################################################################

sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'

if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
########################################################################
sDatabaseName=sDataBaseDir + '/clark.db'

conn = sq.connect(sDatabaseName)
########################################################################

### Import Financial Data
########################################################################

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

sInputFileName

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

print('Loading :',sFileName)

print('##################################')
FinancialRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1") FinancialData=FinancialRawData

print('Loaded Company :',FinancialData.columns.values)
print('##################################')
########################################################################
print('################')

sTable='Assess-Financials'

print('Storing :',sDatabaseName,' Table:',sTable)
```

```
FinancialData.to_sql(sTable, conn, if_exists="replace")

print('#################')
#########################################################################
print(FinancialData.head())

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

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

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

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

```
>>>
========= RESTART: C:\VKHCG\04-Clark\02-Assess\Assess-Financials.py =========
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/01-R/Retrieve_Profit_And_Loss.csv
################################
Loaded Company : ['QTR' 'TypeOfEntry' 'ProductClass1' 'ProductClass2' 'ProductClass3'
 'Amount' 'QTY']
################################
################
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess-Financials
################
       QTR    TypeOfEntry ProductClass1  ... ProductClass3    Amount       QTY
0  2017Q02  Cost of Sales   Hot Blanket  ...           NaN   2989.20   12000.0
1  2017Q02  Cost of Sales     Kitty Box  ...           NaN  19928.00   30000.0
2  2017Q02  Cost of Sales      Maxi Dog  ...           NaN  34874.00   15000.0
3  2017Q02  Cost of Sales     Muis Huis  ...           NaN  29892.00    4000.0
4  2017Q02  Cost of Sales     Water Jug  ...           NaN    199.28  180000.0

[5 rows x 7 columns]
################################
Rows :  2442
################################
### Done!! ############################################
>>>
```

5M. Write a Python program to generate payroll from the given data

```
import sys

import os

import sqlite3 as sq

import pandas as pd
#####################################################################
Base='C:/VKHCG'

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

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

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

Company='04-Clark' sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve-
Data_female-names.csv' sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve-
Data_male-names.csv' sInputFileName3='01-Retrieve/01-EDS/02-Python/Retrieve-
Data_last-names.csv' sOutputFileName1='Assess-Staff.csv' sOutputFileName2='Assess-
Customers.csv'
#####################################################################

sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'

if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/clark.db'

conn = sq.connect(sDatabaseName)
##################################################################### ###

Import Female Data
#####################################################################
sFileName=Base + '/' + Company + '/' +

sInputFileName1 print('#################################')

print('Loading :',sFileName)

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

print(sFileName) FemaleRawData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")

FemaleRawData.rename(columns={'NameValues' : 'FirstName'},inplace=True)
FemaleRawData.drop_duplicates(subset=None, keep='first', inplace=True)
FemaleData=FemaleRawData.sample(100)

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

```
sTable='Assess_FemaleName'

print('Storing :',sDatabaseName,' Table:',sTable)


FemaleData.to_sql(sTable, conn, if_exists="replace")

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

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

print('Rows : ',FemaleData.shape[0], ' records')

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

Import Male Data sFileName=Base + '/' + Company + '/' + sInputFileName2
print('###################################')

print('Loading :',sFileName)

print('###################################')
MaleRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
MaleRawData.rename(columns={'NameValues' : 'FirstName'},inplace=True)
MaleRawData.drop_duplicates(subset=None, keep='first', inplace=True)
MaleData=MaleRawData.sample(100)

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

sTable='Assess_MaleName'

print('Storing :',sDatabaseName,' Table:',sTable)

MaleData.to_sql(sTable, conn, if_exists="replace")

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

print('Rows : ',MaleData.shape[0], ' records')

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

Import Surname Data sFileName=Base + '/' + Company + '/' + sInputFileName3
print('###################################')

print('Loading :',sFileName)

print('###################################')
SurnameRawData=pd.read_csv(sFileName,header=0,

low_memory=False, encoding="latin-1")

SurnameRawData.rename(columns={'NameValues' : 'LastName'},inplace=True)
SurnameRawData.drop_duplicates(subset=None, keep='first', inplace=True)
SurnameData=SurnameRawData.sample(200)

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

```
sTable='Assess_Surname'
print('Storing :',sDatabaseName,' Table:',sTable)
SurnameData.to_sql(sTable, conn, if_exists="replace")
print('#################')
print('###################################')
print('Rows : ',SurnameData.shape[0], ' records')
print('###################################')
print('#################')
sTable='Assess_FemaleName & Assess_MaleName'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " 'Female' as Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FemaleName as A"
sSQL=sSQL+ " UNION"
sSQL=sSQL+ " select distinct"
sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " 'Male' as Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_MaleName as A;"
FirstNameData=pd.read_sql_query(sSQL, conn)
print('#################')
#########################################################################
#print('#################')
sTable='Assess_FirstName'
print('Storing :',sDatabaseName,' Table:',sTable)
FirstNameData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#########################################################################
#########################################################################
print('#################')
sTable='Assess_FirstName x2 & Assess_Surname'
```

```
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " B.FirstName AS SecondName,"
sSQL=sSQL+ " C.LastName,"
sSQL=sSQL+ " A.Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FirstName as A"
 sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_FirstName as B"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Surname as C"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " A.Gender = B.Gender"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " A.FirstName <> B.FirstName;"
PeopleRawData=pd.read_sql_query(sSQL, conn)
People1Data=PeopleRawData.sample(10000)
sTable='Assess_FirstName & Assess_Surname'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct" sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " " AS SecondName,"
sSQL=sSQL+ " B.LastName,"
sSQL=sSQL+ " A.Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FirstName as A"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Surname as B;"
PeopleRawData=pd.read_sql_query(sSQL, conn)
People2Data=PeopleRawData.sample(10000)
```

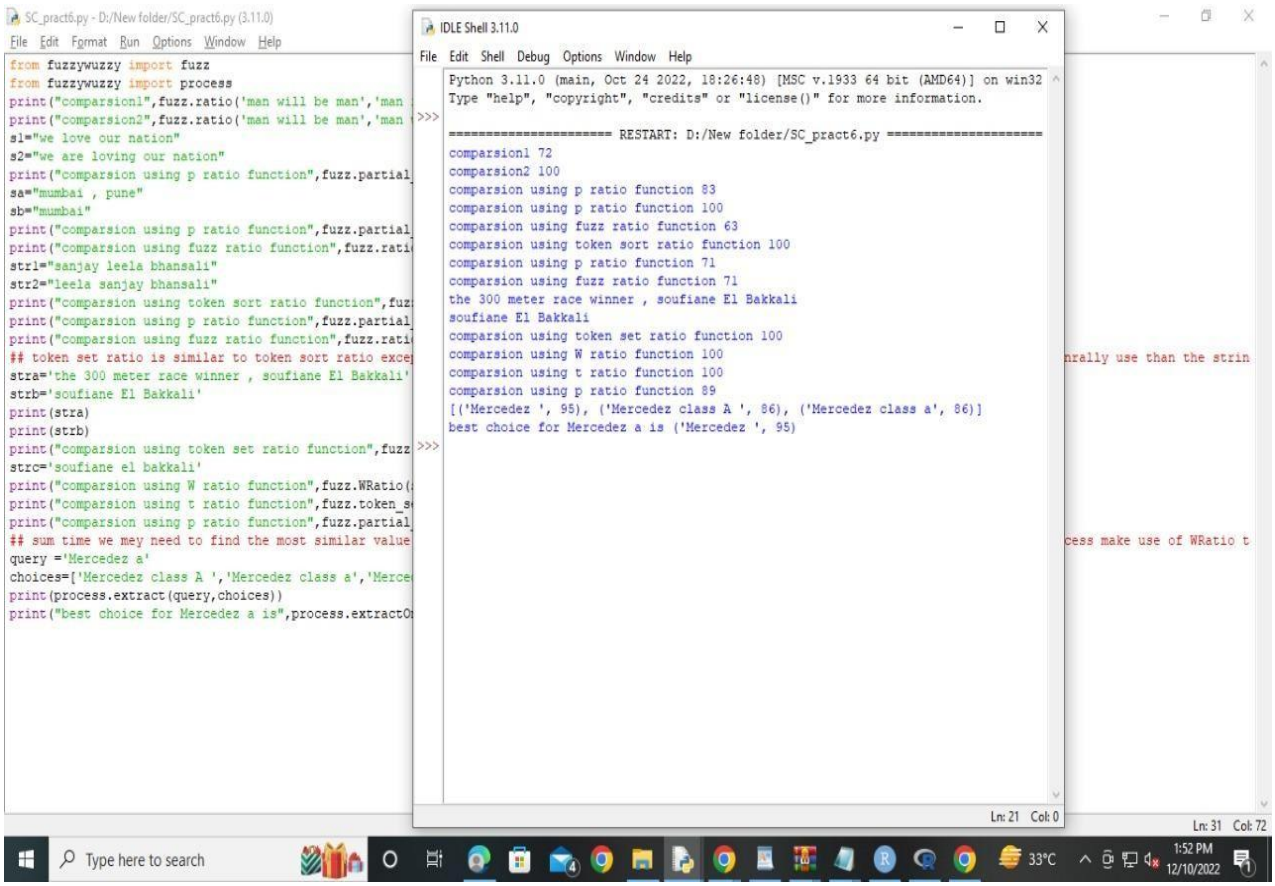
```
PeopleData=People1Data.append(People2Data)

print(PeopleData)

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

sTable='Assess_People'

print('Storing :',sDatabaseName,' Table:',sTable)

PeopleData.to_sql(sTable, conn, if_exists="replace")

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not
os.path.exists(sFileDir): os.makedirs(sFileDir)
#########################################################################
sOutputFileName = sTable+'.csv'

sFileName=sFileDir + '/' +

sOutputFileName

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

print('Storing :', sFileName)

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

PeopleData.to_csv(sFileName, index = False)

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

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

OUTPUT:

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

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

Practical:6

Clustering.




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

```
import uuid pd.options.mode.chained_assignment = None
#####################################################################

if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG' else: Base='C:/VKHCG'
print('###################################')

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

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

 Company='03-Hillman' InputDir='00-RawData'

InputFileName='VehicleData.csv'
#####################################################################

sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite' if not
os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'

conn1 = sq.connect(sDatabaseName)
#####################################################################

sDataVaultDir=Base + '/88-DV' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'

conn2 = sq.connect(sDatabaseName)
#####################################################################

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")

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") t+=1 z+=1
IDZoneNumber=str(uuid.uuid4())

TimeZoneLine=[('ZoneBaseKey', ['UTC']),

('IDZoneNumber', [IDZoneNumber]), ('DateTimeKey', [DateTimeKey]),

('UTCDateTimeValue', [sDateTime]), ('Zone', [zone]), ('DateTimeValue', [sZoneDateTime])]
if t==1: TimeZoneFrame = pd.DataFrame.from_items(TimeZoneLine)

else: TimeZoneRow = pd.DataFrame.from_items(TimeZoneLine) TimeZoneFrame =
TimeZoneFrame.append(TimeZoneRow)
TimeZoneFrameIndex=TimeZoneFrame.set_index(['IDZoneNumber'], inplace=False)
sZone=zone.replace('/','-').replace(' ','')
######################################################################

sTable = 'Process-Time-'+sZone

print('Storing :',sDatabaseName,' Table:',sTable)

TimeZoneFrameIndex.to_sql(sTable, conn1, if_exists="replace")
######################################################################
######################################################################
```

sTable = 'Satellite-Time-'+sZone print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn2, if_exists="replace")

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

print('Vacuum Databases')

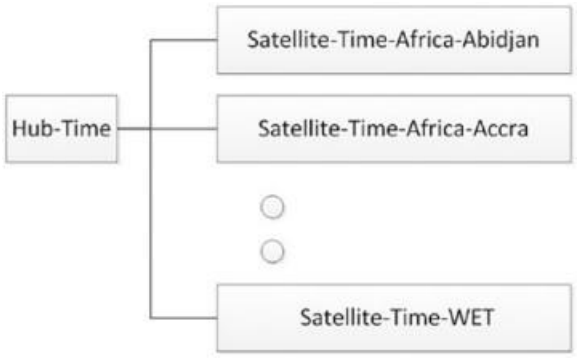
sSQL="VACUUM;"

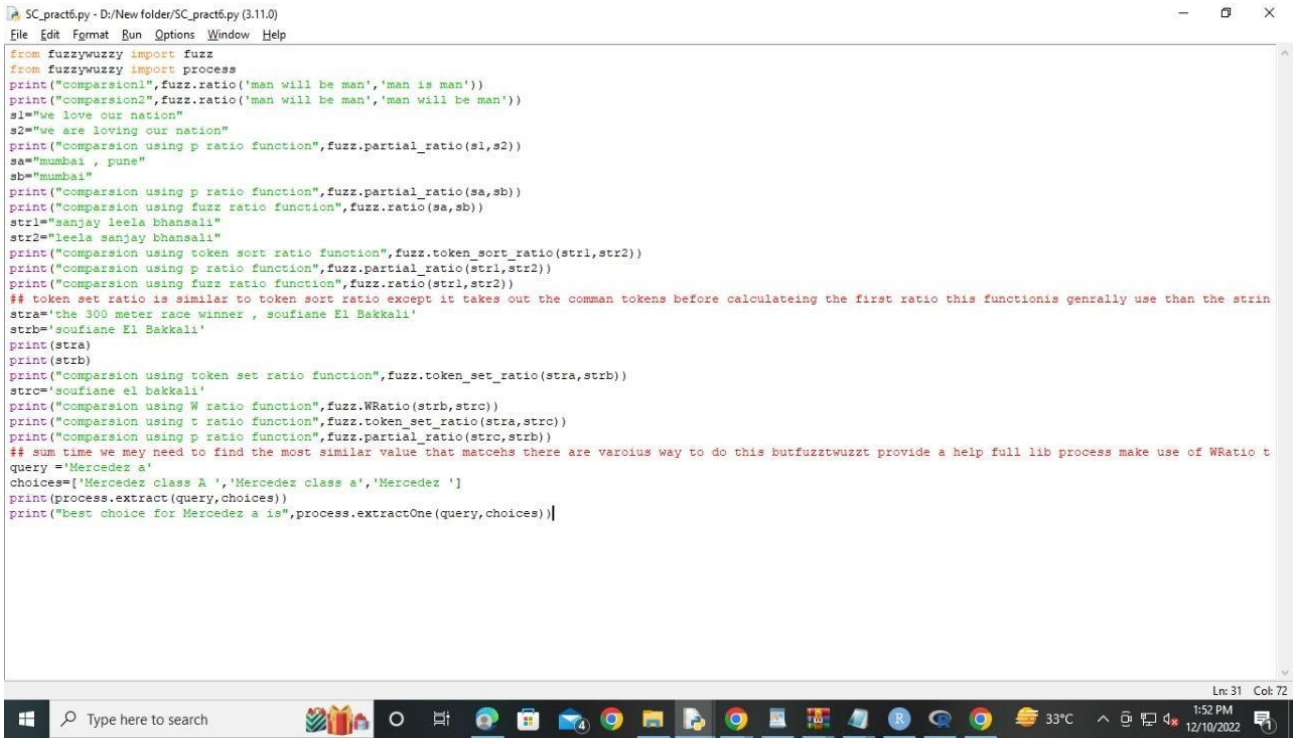
sql.execute(sSQL,conn1)

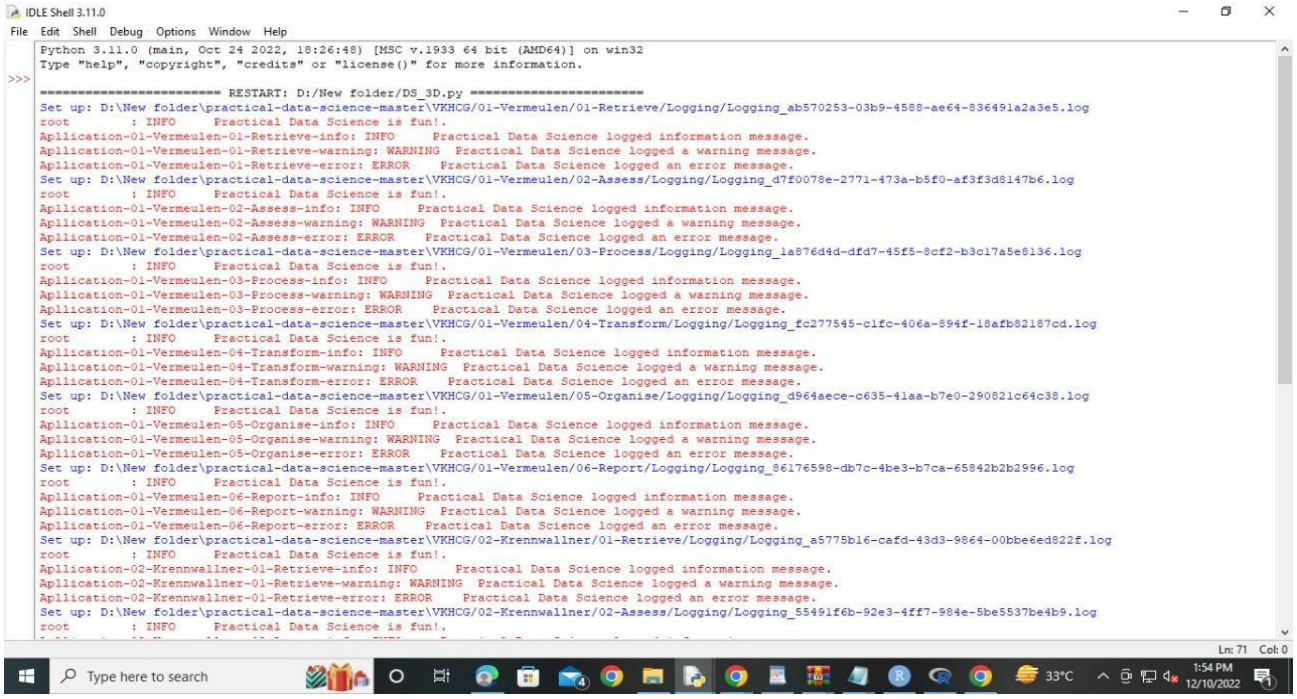
sql.execute(sSQL,conn2)

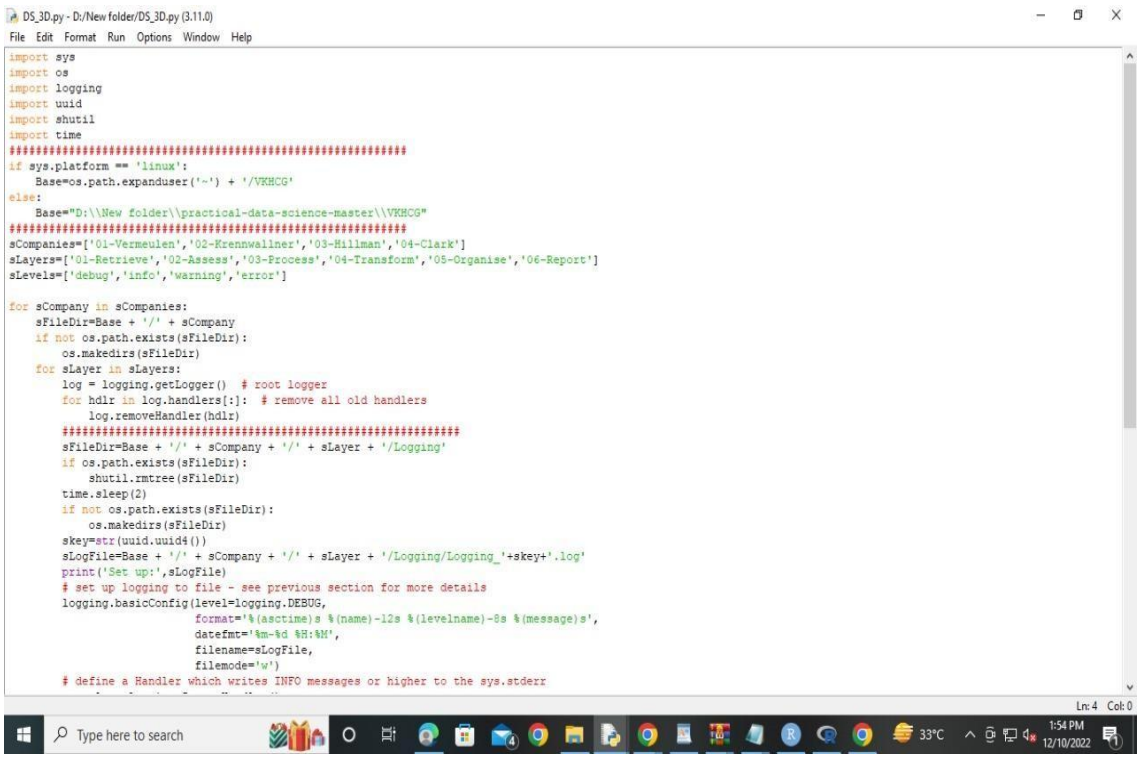
print('#################')
##

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






IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/New folder/DS_3D.py ==========
Set up: D:\New folder\practical-data-science-master\VKHCG/01-Vermeulen/01-Retrieve/Logging/Logging_ab570253-03b9-4588-ae64-836491a2a3e5.log
root : INFO Practical Data Science is fun!.
Application-01-Vermeulen-01-Retrieve-info: INFO Practical Data Science logged information message.
Application-01-Vermeulen-01-Retrieve-warning: WARNING Practical Data Science logged a warning message.
Application-01-Vermeulen-01-Retrieve-error: ERROR Practical Data Science logged an error message.
Set up: D:\New folder\practical-data-science-master\VKHCG/01-Vermeulen/02-Assess/Logging/Logging_d7f0078e-2771-473a-b5f0-af3f3d8147b6.log
root : INFO Practical Data Science is fun!.
Application-01-Vermeulen-02-Assess-info: INFO Practical Data Science logged information message.
Application-01-Vermeulen-02-Assess-warning: WARNING Practical Data Science logged a warning message.
Application-01-Vermeulen-02-Assess-error: ERROR Practical Data Science logged an error message.
Set up: D:\New folder\practical-data-science-master\VKHCG/01-Vermeulen/03-Process/Logging/Logging_1a876d4d-dfd7-45f5-8cf2-b3c17a5e8136.log
root : INFO Practical Data Science is fun!.
Application-01-Vermeulen-03-Process-info: INFO Practical Data Science logged information message.
Application-01-Vermeulen-03-Process-warning: WARNING Practical Data Science logged a warning message.
Application-01-Vermeulen-03-Process-error: ERROR Practical Data Science logged an error message.
Set up: D:\New folder\practical-data-science-master\VKHCG/01-Vermeulen/04-Transform/Logging/Logging_fc277545-c1fc-406a-894f-18afb82187cd.log
root : INFO Practical Data Science is fun!.
Application-01-Vermeulen-04-Transform-info: INFO Practical Data Science logged information message.
Application-01-Vermeulen-04-Transform-warning: WARNING Practical Data Science logged a warning message.
Application-01-Vermeulen-04-Transform-error: ERROR Practical Data Science logged an error message.
Set up: D:\New folder\practical-data-science-master\VKHCG/01-Vermeulen/05-Organise/Logging/Logging_d964aece-c635-41aa-b7e0-290821c64c38.log
root : INFO Practical Data Science is fun!.
Application-01-Vermeulen-05-Organise-info: INFO Practical Data Science logged information message.
Application-01-Vermeulen-05-Organise-warning: WARNING Practical Data Science logged a warning message.
Application-01-Vermeulen-05-Organise-error: ERROR Practical Data Science logged an error message.
Set up: D:\New folder\practical-data-science-master\VKHCG/01-Vermeulen/06-Report/Logging/Logging_86176598-db7c-4be3-b7ca-65842b2b2996.log
root : INFO Practical Data Science is fun!.
Application-01-Vermeulen-06-Report-info: INFO Practical Data Science logged information message.
Application-01-Vermeulen-06-Report-warning: WARNING Practical Data Science logged a warning message.
Application-01-Vermeulen-06-Report-error: ERROR Practical Data Science logged an error message.
Set up: D:\New folder\practical-data-science-master\VKHCG/02-Krennwallner/01-Retrieve/Logging/Logging_a5775b16-cafd-43d3-9864-00bbe6ed822f.log
root : INFO Practical Data Science is fun!.
Application-02-Krennwallner-01-Retrieve-info: INFO Practical Data Science logged information message.
Application-02-Krennwallner-01-Retrieve-warning: WARNING Practical Data Science logged a warning message.
Application-02-Krennwallner-01-Retrieve-error: ERROR Practical Data Science logged an error message.
Set up: D:\New folder\practical-data-science-master\VKHCG/02-Krennwallner/02-Assess/Logging/Logging_55491f6b-92e3-4ff7-984e-5be5537be4b9.log
root : INFO Practical Data Science is fun!.
Ln: 71 Col: 0
Type here to search
33°C
1:54 PM
12/10/2022


DS_3D.py - D:/New folder/DS_3D.py (3.11.0)
File Edit Format Run Options Window Help
import sys
import os
import logging
import uuid
import shutil
import time
###
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base="D:\\New folder\\practical-data-science-master\\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
Ln: 4 Col: 0
Type here to search
33°C
1:54 PM
12/10/2022

```
import sys
import os
import sqlite3 as sq
import pandas as pd from pandas.io
import sql from datetime import datetime, timedelta from pytz
import timezone, all_timezones from random
import randint i
mport uuid ######################################################################
if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else: Base='C:/VKHCG' print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
######################################################################
Company='04-Clark'
sInputFileName='02-Assess/01-EDS/02-Python/Assess_People.csv'
######################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
######################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn1 = sq.connect(sDatabaseName)
######################################################################
sDataVaultDir=Base + '/88-DV' if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
######################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
###################################################################### ###
Import Female Data
######################################################################
sFileName=Base + '/' + Company + '/' +
sInputFileName
print('###################################')
print('Loading :',sFileName)
print('###################################')
print(sFileName) RawData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
```

```
RawData.drop_duplicates(subset=None, keep='first', inplace=True)

start_date = datetime(1900,1,1,0,0,0)

start_date_utc=start_date.replace(tzinfo=timezone('UTC'))

HoursBirth=100*365*24 RawData['BirthDateUTC']=RawData.apply(lambda row:
(start_date_utc + timedelta(hours=randint(0, HoursBirth))) ,axis=1)

zonemax=len(all_timezones)-1

RawData['TimeZone']=RawData.apply(lambda row: (all_timezones[randint(0, zonemax)])
,axis=1) RawData['BirthDateISO']=RawData.apply(lambda row:
row["BirthDateUTC"].astimezone(timezone(row['TimeZone'])) ,axis=1)
RawData['BirthDateKey']=RawData.apply(lambda row:
row["BirthDateUTC"].strftime("%Y-%m-%d %H:%M:%S") ,axis=1)

RawData['BirthDate']=RawData.apply(lambda row: row["BirthDateISO"].strftime("%Y-%m-
%d %H:%M:%S") ,axis=1)

RawData['PersonID']=RawData.apply(lambda row: str(uuid.uuid4()) ,axis=1)
#########################################################################

Data=RawData.copy() Data.drop('BirthDateUTC', axis=1,inplace=True)

Data.drop('BirthDateISO', axis=1,inplace=True)

indexed_data = Data.set_index(['PersonID'])

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

sTable='Process_Person'

print('Storing :',sDatabaseName,' Table:',sTable)

indexed_data.to_sql(sTable, conn1, if_exists="replace")

print('#################')
#########################################################################
PersonHubRaw=Data[['PersonID','FirstName','SecondName','LastName','BirthDateKey']]
PersonHubRaw['PersonHubID']=RawData.apply(lambda row: str(uuid.uuid4()) ,axis=1)
PersonHub=PersonHubRaw.drop_duplicates(subset=None, \ keep='first',\ inplace=False)
indexed_PersonHub = PersonHub.set_index(['PersonHubID'])

sTable = 'Hub-Person' print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonHub.to_sql(sTable, conn2, if_exists="replace")

PersonSatelliteGenderRaw=Data[['PersonID','FirstName','SecondName','LastName'\
,'BirthDateKey','Gender']]
PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lambda row:
str(uuid.uuid4()) ,axis=1)
PersonSatelliteGender=PersonSatelliteGenderRaw.drop_duplicates(subset=None, \
keep='first', \ inplace=False) indexed_PersonSatelliteGender =
PersonSatelliteGender.set_index(['PersonSatelliteID']) sTable = 'Satellite-Person-Gender'
print('Storing :',sDatabaseName,' Table:',sTable)
```

```
indexed_PersonSatelliteGender.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
PersonSatelliteBirthdayRaw=Data[['PersonID','FirstName','SecondName','LastName',\
'BirthDateKey','TimeZone','BirthDate']]
PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lambda row:
str(uuid.uuid4()) ,axis=1)

PersonSatelliteBirthday=PersonSatelliteBirthdayRaw.drop_duplicates(subset=None, \
keep='first',\ inplace=False) indexed_PersonSatelliteBirthday =
PersonSatelliteBirthday.set_index(['PersonSatelliteID'])

sTable = 'Satellite-Person-Names'

print('Storing :',sDatabaseName,' Table:',sTable)

indexed_PersonSatelliteBirthday.to_sql(sTable, conn2,

if_exists="replace")
#####################################################################
sFileDir=Base + '/' + Company + '/03-Process/01-EDS/02-Python' if not
os.path.exists(sFileDir): os.makedirs(sFileDir)

#####################################################################
sOutputFileName = sTable + '.csv' sFileName=sFileDir + '/' + sOutputFileName
print('###################################')

print('Storing :', sFileName)

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

RawData.to_csv(sFileName, index = False)

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

print('Vacuum Databases')

sSQL="VACUUM;"

sql.execute(sSQL,conn1)

sql.execute(sSQL,conn2)

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

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

Load the person into the data vault

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

```
import sys
import os
import pandas as pd
import sqlite3 as sq from pandas.io
import sql
import uuid pd.options.mode.chained_assignment = None
################################################################
if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else: Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
Company='03-Hillman' InputDir='00-RawData' InputFileName='VehicleData.csv'
################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite' if not
os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
```

```
#################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'

conn1 = sq.connect(sDatabaseName)
#################################################################

sDataVaultDir=Base + '/88-DV'

if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'

conn2 = sq.connect(sDatabaseName)
#################################################################

sFileName=Base + '/' + Company + '/' + InputDir + '/' +

InputFileName

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

print('Loading :',sFileName)

VehicleRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#################################################################
sTable='Process_Vehicles'

print('Storing :',sDatabaseName,' Table:',sTable)

VehicleRaw.to_sql(sTable, conn1, if_exists="replace")
#################################################################
VehicleRawKey=VehicleRaw[['Make','Model']].copy()

VehicleKey=VehicleRawKey.drop_duplicates()
#################################################################
VehicleKey['ObjectKey']=VehicleKey.apply(lambda row: str('('+
str(row['Make']).strip().replace(' ', '-').replace('/', '-').lower() + ')-(' +
(str(row['Model']).strip().replace(' ', '-').replace(' ', '-').lower()) +')') ,axis=1)
#################################################################
VehicleKey['ObjectType']=VehicleKey.apply(lambda row: 'vehicle' ,axis=1)
#################################################################
VehicleKey['ObjectUUID']=VehicleKey.apply(lambda row: str(uuid.uuid4()) ,axis=1)

VehicleHub=VehicleKey[['ObjectType','ObjectKey','ObjectUUID']].copy()
VehicleHub.index.name='ObjectHubID' sTable = 'Hub-Object-Vehicle' print('Storing
:',sDatabaseName,' Table:',sTable)

VehicleHub.to_sql(sTable, conn2, if_exists="replace")
################################################################# ### Vehicle
Satellite ################################################################# #
VehicleSatellite=VehicleKey[['ObjectType','ObjectKey','ObjectUUID','Make','Model']].copy(
) VehicleSatellite.index.name='ObjectSatelliteID' sTable = 'Satellite-Object-Make-Model'

print('Storing :',sDatabaseName,' Table:',sTable)
```

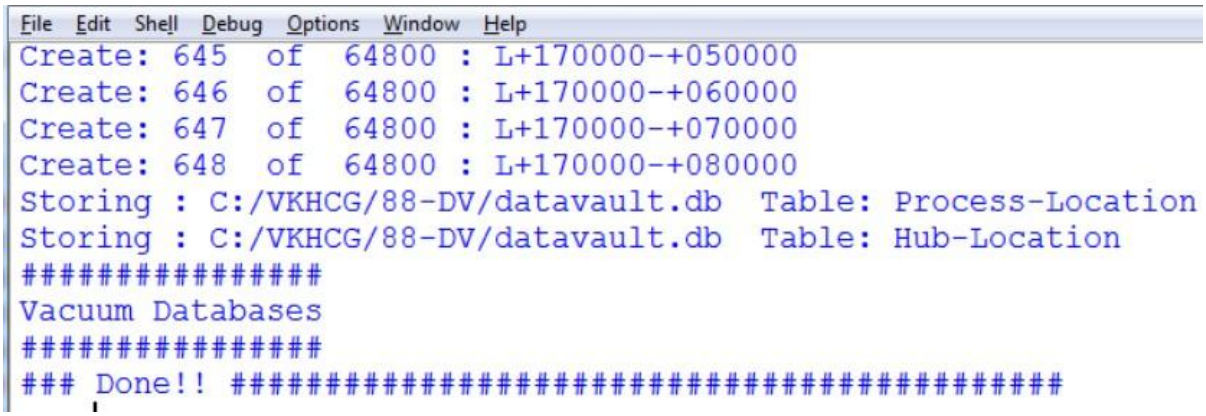
```
VehicleSatellite.to_sql(sTable, conn2, if_exists="replace")
#########################################################################
### Vehicle Dimension
#########################################################################
sView='Dim-Object' print('Storing :',sDatabaseName,' View:',sView)
sSQL="CREATE VIEW IF NOT EXISTS [" + sView + "] AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " H.ObjectType,"
sSQL=sSQL+ " H.ObjectKey AS VehicleKey,"
sSQL=sSQL+ " TRIM(S.Make) AS VehicleMake,"
sSQL=sSQL+ " TRIM(S.Model) AS VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " [Hub-Object-Vehicle] AS H"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " [Satellite-Object-Make-Model] AS S"
sSQL=sSQL+ " ON"
sSQL=sSQL+ " H.ObjectType=S.ObjectType"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " H.ObjectUUID=S.ObjectUUID;"
sql.execute(sSQL,conn2)
print('##################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " VehicleMake,"
sSQL=sSQL+ " VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " [" + sView + "]"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " VehicleMake"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " VehicleMake;"
DimObjectData=pd.read_sql_query(sSQL, conn2)
 DimObjectData.index.name='ObjectDimID'
DimObjectData.sort_values(['VehicleMake','VehicleModel'],inplace=True, ascending=True)
print('##################')
```

```
print(DimObjectData)
################################################################################
print('################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('################')
################################################################################
conn1.close()
conn2.close()
################################################################################ print('### Done!! ############################################')
```

```
RESTART: C:\VKHCG\03-Hillman\05-Process\Process-Vehicle-Logistics.py
################################
Working Base : C:/VKHCG  using  win32
################################
###########
Loading : C:/VKHCG/03-Hillman/00-RawData/VehicleData.csv
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process_Vehicles
Storing : C:/VKHCG/88-DV/datavault.db  Table: Hub-Object-Vehicle
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Object-Make-Model
Storing : C:/VKHCG/88-DV/datavault.db  View: Dim-Object
################
Loading : C:/VKHCG/88-DV/datavault.db  Table: Dim-Object
################
                  VehicleMake                        VehicleModel
ObjectDimID
2213               AM General                   DJ Po Vehicle 2WD
2212               AM General                    FJ8c Post Office
129                AM General                 Post Office DJ5 2WD
131                AM General                 Post Office DJ8 2WD
2869          ASC Incorporated                                GNX
...                       ...                                 ...
1996                    smart                  fortwo convertible
1997                    smart                        fortwo coupe
2622                    smart      fortwo electric drive cabriolet
2833                    smart    fortwo electric drive convertible
2623                    smart          fortwo electric drive coupe

[3885 rows x 2 columns]
################
Vacuum Databases
################
```

```
import sys

import os

import pandas as pd

import sqlite3 as sq from pandas.io

import sql

import uuid ########################################################################
Base='C:/VKHCG'

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

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

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

Company='01-Vermeulen'

InputAssessGraphName='Assess_All_Animals.gml'

EDSAssessDir='02-Assess/01-EDS' InputAssessDir=EDSAssessDir + '/02-Python'
######################################################################## sFileAssessDir=Base +
'/' + Company + '/' + InputAssessDir if not os.path.exists(sFileAssessDir): os.makedirs(sFileAssessDir)
######################################################################## sDataBaseDir=Base +
'/' + Company + '/03-Process/SQLite' if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
########################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'

conn1 = sq.connect(sDatabaseName)
########################################################################

sDataVaultDir=Base + '/88-DV' if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
########################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'

conn2 = sq.connect(sDatabaseName)

t=0

tMax=360*180

for Longitude in range(-180,180,10): for Latitude in range(-90,90,10):

t+=1

IDNumber=str(uuid.uuid4())

LocationName='L'+format(round(Longitude,3)*1000, '+07d') +\ '-'+format(round(Latitude,3)*1000,
'+07d') print('Create:',t,' of ',tMax,':',LocationName)

LocationLine=[('ObjectBaseKey', ['GPS']), ('IDNumber', [IDNumber]), ('LocationNumber', [str(t)]),
('LocationName', [LocationName]), ('Longitude', [Longitude]), ('Latitude', [Latitude])] if t==1:
LocationFrame = pd.DataFrame.from_items(LocationLine)

else: LocationRow = pd.DataFrame.from_items(LocationLine) LocationFrame =
LocationFrame.append(LocationRow)
########################################################################
```

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



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

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

```
print('Rows :',ShareData.shape[0])
print('Columns:',ShareData.shape[1])
print('################')
#########################################################################
print('################')
sTable=str(RawData['sTable'][sShare])
print('Storing :',sDatabaseName,' Table:',sTable)
ShareData.to_sql(sTable, conn, if_exists="replace")
print('################')
#########################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
ShareData.to_csv(sFileName, index = False)
print('################################')
#########################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir2 + '/Assess_' +
sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
ShareData.to_csv(sFileName, index = False)
print('################################')
#########################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir3 + '/Process_' +
sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
ShareData.to_csv(sFileName, index = False)
print('################################')
```

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

Output:

```
======= RESTART: C:\VKHCG\04-Clark\03-Process\Process-Shares-Data.py =======
Working Base : C:/VKHCG using win32
Loading : C:/VKHCG/04-Clark/00-RawData/VKHCG_Shares.csv
Rows : 10
Columns: 3
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_Shares.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_Shares.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_Shares.csv
Share : WIKI/GOOGL
Rows : 3424
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Google
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Google.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Google.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Google.csv
Share : WIKI/MSFT
Rows : 8076
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Microsoft
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Microsoft.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Microsoft.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Microsoft.csv
Share : WIKI/UPS
Rows : 4622
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_UPS
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_UPS.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_UPS.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_UPS.csv
Share : WIKI/AMZN
Rows : 5248
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Amazon
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Amazon.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Amazon.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Amazon.csv
Share : LOCALBTC/USD
Rows : 1863
Columns: 6
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: LOCALBTC_USD
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_LOCALBTC_USD.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_LOCALBTC_USD.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_LOCALBTC_USD.csv
Share : PERTH/AUD_USD_M
Rows : 340
Columns: 8
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: PERTH_AUD_USD_M
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_PERTH_AUD_USD_M.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_PERTH_AUD_USD_M.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_PERTH_AUD_USD_M.csv
Share : PERTH/AUD_USD_D
Rows : 7989
Columns: 8
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: PERTH_AUD_USD_D
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_PERTH_AUD_USD_D.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_PERTH_AUD_USD_D.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_PERTH_AUD_USD_D.csv
Share : FRED/GDP
Rows : 290
Columns: 3
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: FRED/GDP
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_FRED-GDP.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_FRED-GDP.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_FRED-GDP.csv
Share : FED/RXI_US_N_A_UK
Rows : 49
Columns: 3
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: FED_RXI_US_N_A_UK
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_FED_RXI_US_N_A_UK.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_FED_RXI_US_N_A_UK.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_FED_RXI_US_N_A_UK.csv
Share : FED/RXI_N_A_CA
Rows : 49
Columns: 3
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: FED_RXI_N_A_CA
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_FED_RXI_N_A_CA.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_FED_RXI_N_A_CA.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_FED_RXI_N_A_CA.csv
```

Practical 7: Transforming Data

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

```
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)") print(BirthDateZoneUTCStr)

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

print('Birth Date in Reykjavik :')

BirthZone = 'Atlantic/Reykjavik'

BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")

print(BirthDateStr)

print('###################################')
#####################################################################
IDZoneNumber=str(uuid.uuid4())

sDateTimeKey=BirthDateZoneStr.replace(' ','-').replace(':','-')

TimeLine=[('ZoneBaseKey', ['UTC']), ('IDNumber', [IDZoneNumber]), ('DateTimeKey',
[sDateTimeKey]), ('UTCDateTimeValue', [BirthDateZoneUTC]), ('Zone', [BirthZone]),
('DateTimeValue', [BirthDateStr])] TimeFrame = pd.DataFrame.from_items(TimeLine)
#####################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#####################################################################

sTable = 'Hub-Time-Gunnarsson'

print('\n###################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n###################################')

TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")

sTable = 'Dim-Time-Gunnarsson'

TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']]
TimeSatelliteIndex=TimeSatellite.set_index(['IDNumber'],inplace=False)
#####################################################################
BirthZoneFix=BirthZone.replace(' ','-').replace('/','-')

sTable = 'Satellite-Time-' + BirthZoneFix + '-Gunnarsson'
print('\n###################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n###################################')

TimeSatelliteIndex.to_sql(sTable, conn2, if_exists="replace")
```

```
sTable = 'Dim-Time-' + BirthZoneFix + '-Gunnarsson'
TimeSatelliteIndex.to_sql(sTable, conn3, if_exists="replace")
#########################################################################
print('\n#####################################')
print('Person Category')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
print('Name:',FirstName,LastName)
print('Birth Date:',BirthDateLocal)
print('Birth Zone:',BirthZone)
print('UTC Birth Date:',BirthDateZoneStr)
print('#####################################')
#########################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('IDNumber', [IDPersonNumber]), ('FirstName', [FirstName]), ('LastName',
[LastName]), ('Zone', ['UTC']), ('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
#########################################################################
TimeHub=PersonFrame
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#########################################################################
sTable = 'Hub-Person-Gunnarsson'
print('\n#####################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#####################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Person-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
```

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

```
import sys
import os from datetime
import datetime from pytz
import timezone
import pandas as pd
import sqlite3 as sq
import uuid pd.options.mode.chained_assignment = None
#####################################################################
if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG' else:
Base='C:/VKHCG'
print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite' if not
os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir): os.makedirs(sDataWarehousetDir)
#####################################################################
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
print('\n#################################')
print('Time Dimension')
BirthZone = 'Atlantic/Reykjavik'
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
```

```
#####################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [IDTimeNumber]), ('UTCDate', [BirthDateZoneStr]), ('LocalTime', [BirthDateLocal]), ('TimeZone', [BirthZone])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
#####################################################################
DimTime=TimeFrame DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
#####################################################################
sTable = 'Dim-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('\n#################################')
print('Dimension Person')
print('\n#################################')
FirstName = 'Guðmundur' LastName = 'Gunnarsson'
#####################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [IDPersonNumber]), ('FirstName', [FirstName]), ('LastName', [LastName]), ('Zone', ['UTC']), ('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####################################################################
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('\n#################################')
print('Fact - Person - time')
print('\n#################################')
```

```
IDFactNumber=str(uuid.uuid4())

PersonTimeLine=[('IDNumber', [IDFactNumber]), ('IDPersonNumber', [IDPersonNumber]),
('IDTimeNumber', [IDTimeNumber])] PersonTimeFrame =
pd.DataFrame.from_items(PersonTimeLine)
#####################################################################
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
#####################################################################

sTable = 'Fact-Person-Time' print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")

FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
```

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8
Type "help", "copyright", "credits" or "lice
>>>
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform
################################
Working Base : C:/VKHCG  using  win32
################################

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

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

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

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

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

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

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

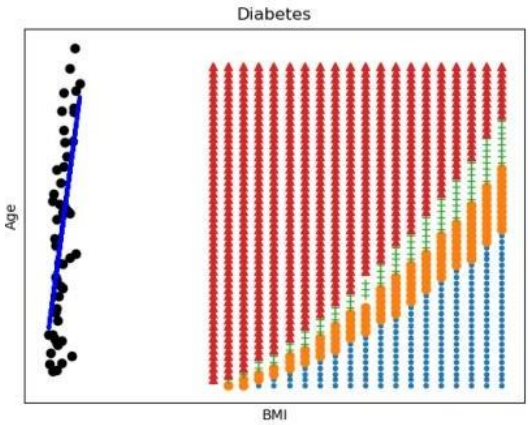
```
bmi = weight/(height*height)
if bmi <= 18.5: BMI_Result=1
elif bmi > 18.5 and bmi < 25: BMI_Result=2
elif bmi > 25 and bmi < 30:
BMI_Result=3 elif bmi > 30:
BMI_Result=4 else:
BMI_Result=0
PersonLine=[('PersonID', [str(t)]), ('Height', [height]), ('Weight', [weight]), ('bmi', [bmi]),
('Indicator', [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()
diabetes = datasets.load_diabetes()
_X = diabetes.data[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-30] diabetes_X_test = diabetes_X[-50:]
diabetes_y_train = diabetes.target[:-30] diabetes_y_test = diabetes.target[-50:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print('Coefficients: \n', regr.coef_)
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test, diabetes_y_pred))
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
```

```
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.axis('tight')
plt.title("Diabetes")
plt.xlabel("BMI")
plt.ylabel("Age")
plt.show()
```

Output:

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

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

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

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

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

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

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

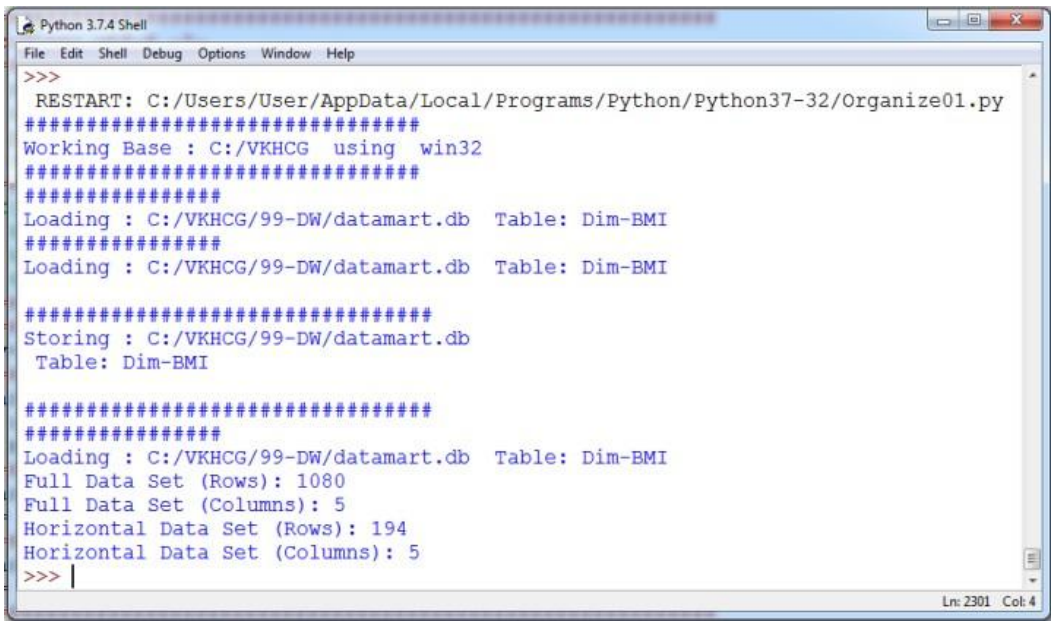


Practical 8: Organizing Data

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

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

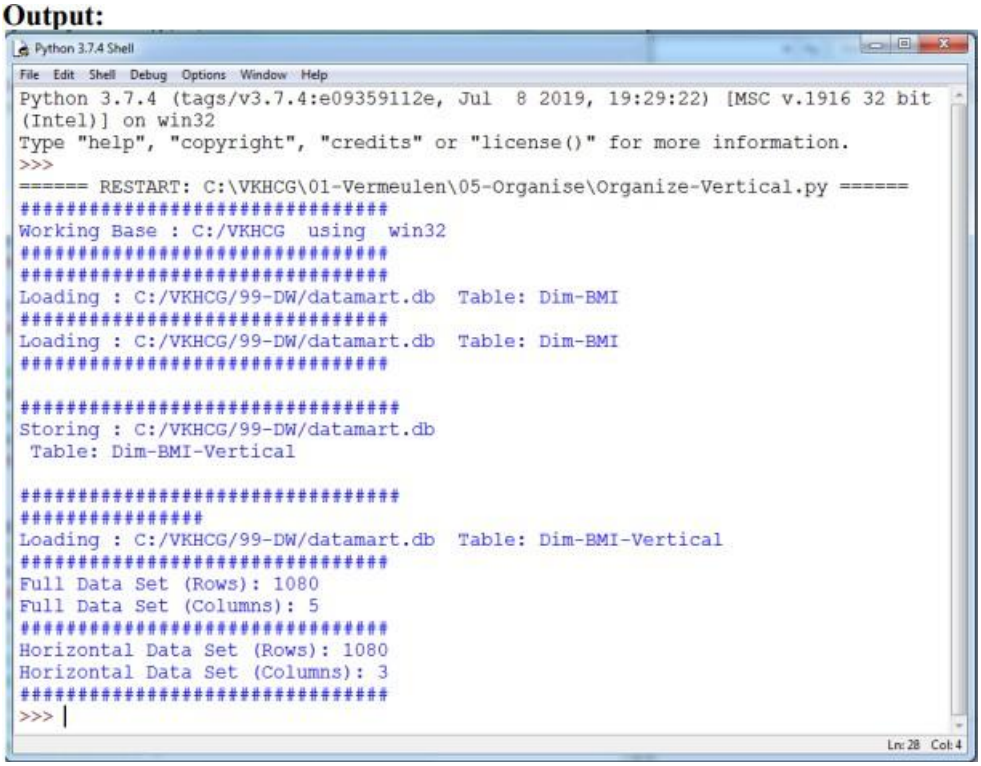
Output:



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

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

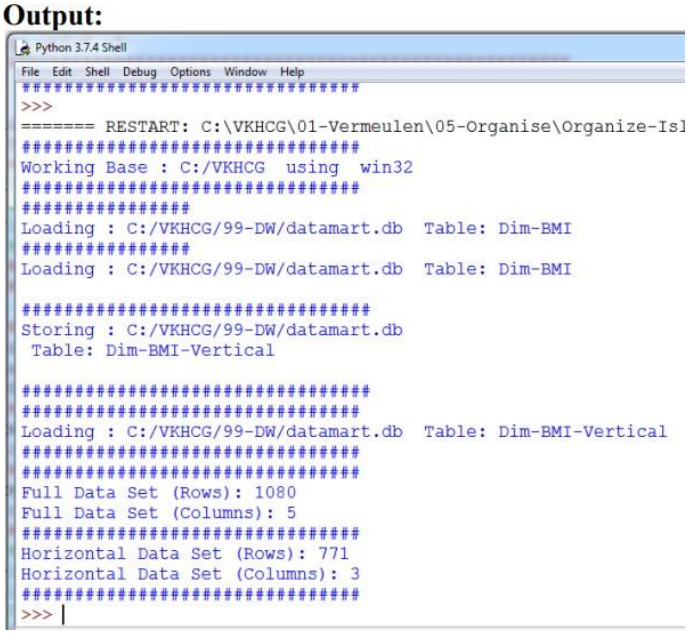
Output:



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

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

Output:



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

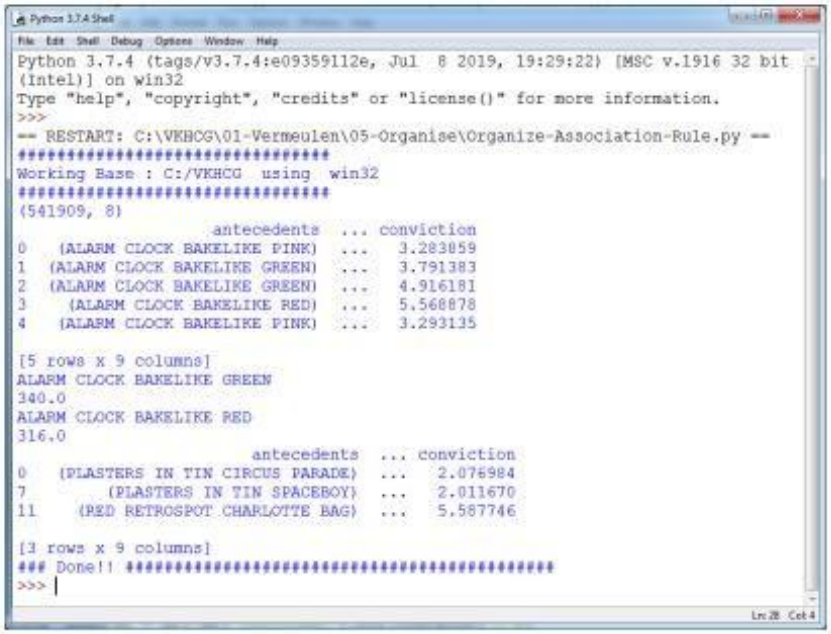
```
sTable = 'Dim-BMI-Secure'
print('\n#####################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#####################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#########################################################################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

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

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

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

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

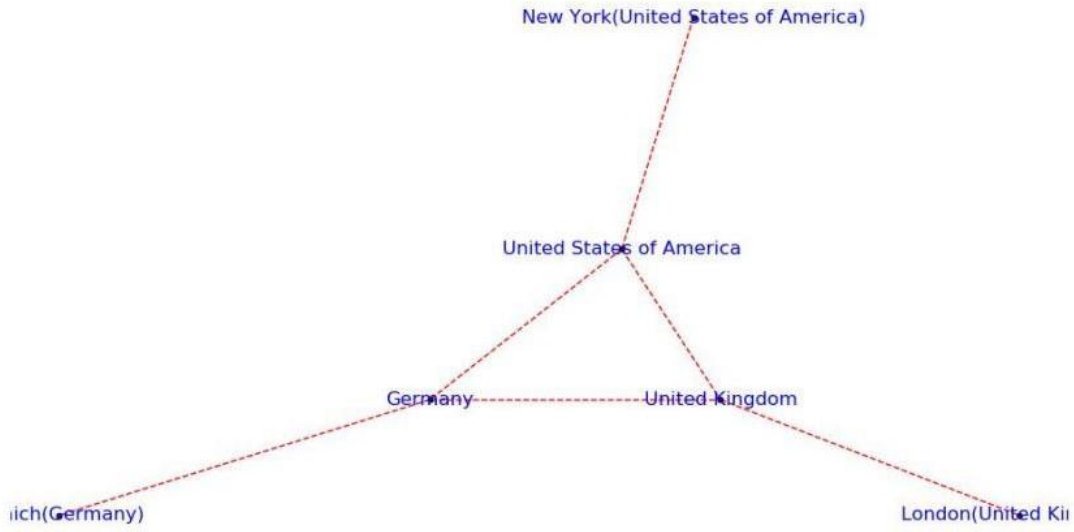
```
rules[ (rules['lift'] >= 6) & (rules['confidence'] >= 0.8) ]
#########################################################################
sProduct1='ALARM CLOCK BAKELIKE GREEN'
print(sProduct1)
print(basket[sProduct1].sum())
sProduct2='ALARM CLOCK BAKELIKE RED'
print(sProduct2)
print(basket[sProduct2].sum())
#########################################################################
basket2 = (df[df['Country'] =="Germany"] .groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0) .set_index('InvoiceNo'))
basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2,
min_support=0.05,
use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)
print(rules2[ (rules2['lift'] >= 4) & (rules2['confidence'] >= 0.5)])
#########################################################################
print('### Done!! ################################################')
```

Output:



```
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
#####################################################################
pd.options.mode.chained_assignment = None
#####################################################################
Base='C:/VKHCG'
#####################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#####################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv'
#####################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Network-Routing-
Company.gml' sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Network-
Routing-Company.png' Company='01-Vermeulen'
#####################################################################
##################################################################### ###
Import Country Data
#####################################################################
sFileName=Base + '/' + Company + '/' +
sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('#################################')
#####################################################################
print(CompanyData.head())
print(CompanyData.shape)
#####################################################################
G=nx.Graph() for i in range(CompanyData.shape[0]):
for j in range(CompanyData.shape[0]):
Node0=CompanyData['Company_Country_Name'][i]
Node1=CompanyData['Company_Country_Name'][j] if
Node0 != Node1: G.add_edge(Node0,Node1)
```

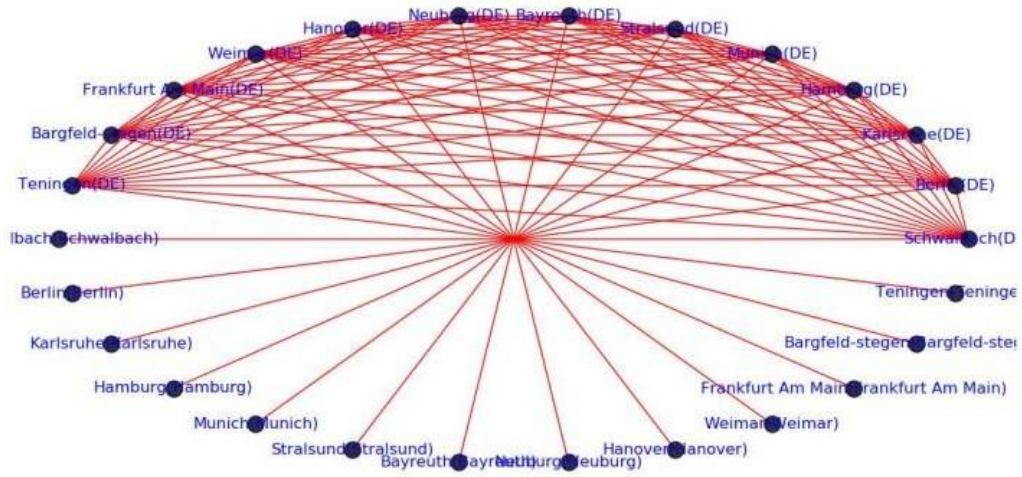
```
for i in range(CompanyData.shape[0]):


Node0=CompanyData['Company_Country_Name'][i]
Node1=CompanyData['Company_Place_Name'][i] + '('+
CompanyData['Company_Country_Name'][i] + ')'
if Node0 != Node1:
G.add_edge(Node0,Node1)
print('Nodes:',
G.number_of_nodes())
print('Edges:', G.number_of_edges())
#########################################################################
sFileName=Base + '/' + Company + '/' +
sOutputFileName1 print('###################################')
print('Storing :',sFileName)
print('###################################')
nx.write_gml(G, sFileName)
#########################################################################
sFileName=Base + '/' + Company + '/' +
sOutputFileName2
print('###################################')
print('Storing Graph Image:',sFileName)
print('###################################')
plt.figure(figsize=(15, 15))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off') plt.savefig(sFileName,dpi=600)
plt.show()
#########################################################################
print('###################################')
print('### Done!! ####################')
print('###################################')
```




```
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
#######################################################################
pd.options.mode.chained_assignment = None
#######################################################################
Base='C:/VKHCG'
#######################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#######################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-DE-Billboard-Visitor.csv'
#######################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Billboards.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Billboards.png'
Company='02-Krennwallner'
#######################################################################
######################################################################## ###
Import Company Data
#######################################################################
sFileName=Base + '/' + Company + '/' +
```

```
sInputFileName
print('##################################')
 print('Loading :',sFileName)
print('##################################')
BillboardDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1") print('##################################')
#######################################################################
print(BillboardDataRaw.head())
print(BillboardDataRaw.shape)
BillboardData=BillboardDataRaw
sSample=list(np.random.choice(BillboardData.shape[0],20))
####################################################################### G=nx.Graph()
for i in sSample:
for j in sSample:
Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['BillboardCountry'][i] +
')' Node1=BillboardData['BillboardPlaceName'][j] + '('+ BillboardData['BillboardCountry'][i]
+ ')' if Node0 != Node1: G.add_edge(Node0,Node1)
for i in sSample:
Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorPlaceName'][i] +
')' Node1=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorCountry'][i] +
')'
if Node0 != Node1: G.add_edge(Node0,Node1)
print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
#######################################################################
sFileName=Base + '/02-Krennwallner/' +
sOutputFileName1
print('##################################')
print('Storing :',sFileName)
print('##################################')
nx.write_gml(G, sFileName)
#######################################################################
sFileName=Base + '/02-Krennwallner/' +
sOutputFileName2
print('##################################')
print('Storing Graph Image:',sFileName)
print('##################################')
```

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




```
import sys
import os
import pandas as pd ###########################################################################
Base='C:/VKHCG' ###############################################################################
print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
#########################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.txt'
#########################################################################
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Routes.csv'
Company='03-Hillman'
sFileName=Base + '/' + Company + '/' +
sInputFileName
```

```
print('#################################')
print('Loading :',sFileName)
print('#################################')
RouteDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, sep='|', encoding="latin-1")
print('#################################')
#####################################################################
RouteStart=RouteDataRaw[RouteDataRaw['StartAt']=='WH-KA13']
#####################################################################
RouteDistance=RouteStart[RouteStart['Cost']=='DistanceMiles']
RouteDistance=RouteDistance.sort_values(by=['Measure'], ascending=False)
#####################################################################
RouteMax=RouteStart["Measure"].max()
RouteMaxCost=round((((RouteMax/1000)*1.5*2)),2) print('#################################')
print('Maximum (£) per day:') print(RouteMaxCost)
print('#################################')
#####################################################################
RouteMean=RouteStart["Measure"].mean()
RouteMeanMonth=round((((RouteMean/1000)*2*30)),6)
print('#################################')
print('Mean per Month (Miles):')
print(RouteMeanMonth) print('#################################')
```

```
======== RESTART: C:\VKHCG\03-Hillman\05-Organise\Organise-Routes.py ========
#################################
Working Base : C:/VKHCG  using  win32
#################################
#################################
Loading : C:/VKHCG/03-Hillman/02-Assess/01-EDS/02-Python/Assess_Shipping_Routes
txt
#################################
#################################
#################################
Maximum (£) per day:
21.82
#################################
#################################
Mean per Month (Miles):
21.56191
#################################
```

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import re ########################################################################
Base='C:/VKHCG'
########################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
########################################################################
sInputFileName='03-Process/01-EDS/02-Python/Process_ExchangeRates.csv'
########################################################################
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Forex.csv'
Company='04-Clark'
########################################################################
sDatabaseName=Base + '/' + Company + '/05-Organise/SQLite/clark.db'
conn = sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
########################################################################
######################################################################## ###
Import Forex Data
########################################################################
sFileName=Base + '/' + Company + '/' +
sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('##################################')
########################################################################
ForexDataRaw.index.names = ['RowID']
sTable='Forex_All' print('Storing :',sDatabaseName,' Table:',sTable)
ForexDataRaw.to_sql(sTable, conn, if_exists="replace")
########################################################################
sSQL="SELECT 1 as Bag\ , CAST(min(Date) AS VARCHAR(10)) as Date \
,CAST(1000000.0000000 as NUMERIC(12,4)) as Money \ ,'USD' as Currency \ FROM
Forex_All \ ;" sSQL=re.sub("\s\s+", " ", sSQL) nMoney=pd.read_sql_query(sSQL, conn)
########################################################################
```

```
nMoney.index.names = ['RowID']
 sTable='MoneyData'
print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
######################################################################
sTable='TransactionData' print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
######################################################################
ForexDay=pd.read_sql_query("SELECT Date FROM Forex_All GROUP BY Date;", conn)
######################################################################
t=0
for i in range(ForexDay.shape[0]):
sDay1=ForexDay['Date'][i]
sDay=str(sDay1)
sSQL='\ SELECT M.Bag as Bag, \
F.Date as Date,
\ round(M.Money * F.Rate,6) AS Money,
\ F.CodeIn AS PCurrency,
\ F.CodeOut AS Currency
\ FROM MoneyData AS M
\ JOIN \ ( \ SELECT \ CodeIn, CodeOut, Date, Rate
\ FROM \ Forex_All \ WHERE\ CodeIn = "USD" AND CodeOut = "GBP"
 \ UNION \ SELECT \ CodeOut AS CodeIn, CodeIn AS CodeOut, Date, (1/Rate) AS Rate
\ FROM \ Forex_All \ WHERE\ CodeIn = "USD" AND CodeOut = "GBP" \ ) AS F
\ ON \ M.Currency=F.CodeIn \ AND \ F.Date ='" +sDay + "';'
sSQL=re.sub("\s\s+", " ", sSQL)
ForexDayRate=pd.read_sql_query(sSQL, conn)
for j in range(ForexDayRate.shape[0]):
sBag=str(ForexDayRate['Bag'][j])
nMoney=str(round(ForexDayRate['Money'][j],2))
sCodeIn=ForexDayRate['PCurrency'][j]
sCodeOut=ForexDayRate['Currency'][j]
sSQL='UPDATE MoneyData SET Date= '" + sDay + "', ' sSQL= sSQL + ' Money = ' +
nMoney + ', Currency='" + sCodeOut + ""
sSQL= sSQL + ' WHERE Bag=' + sBag + ' AND Currency='" + sCodeIn + "';'
sSQL=re.sub("\s\s+", " ", sSQL) cur = conn.cursor()
```

```
cur.execute(sSQL) conn.commit()

t+=1

print('Trade :', t, sDay, sCodeOut, nMoney)

sSQL=' \ INSERT INTO TransactionData ( \ RowID, \

Bag, \ Date, \ Money, \ Currency \ ) \

SELECT ' + str(t) + ' AS RowID, \ Bag, \ Date, \ Money, \ Currency \ FROM MoneyData \ ;'
sSQL=re.sub("\s\s+", " ", sSQL)

cur = conn.cursor()

cur.execute(sSQL)

conn.commit()
#####################################################################
sSQL="SELECT RowID, Bag, Date, Money, Currency FROM TransactionData ORDER BY
RowID;" sSQL=re.sub("\s\s+", " ", sSQL)

TransactionData=pd.read_sql_query(sSQL, conn)

OutputFile=Base + '/' + Company + '/' +

sOutputFileName TransactionData.to_csv(OutputFile, index = False)
```

Practical 9: Generating Data

```
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')

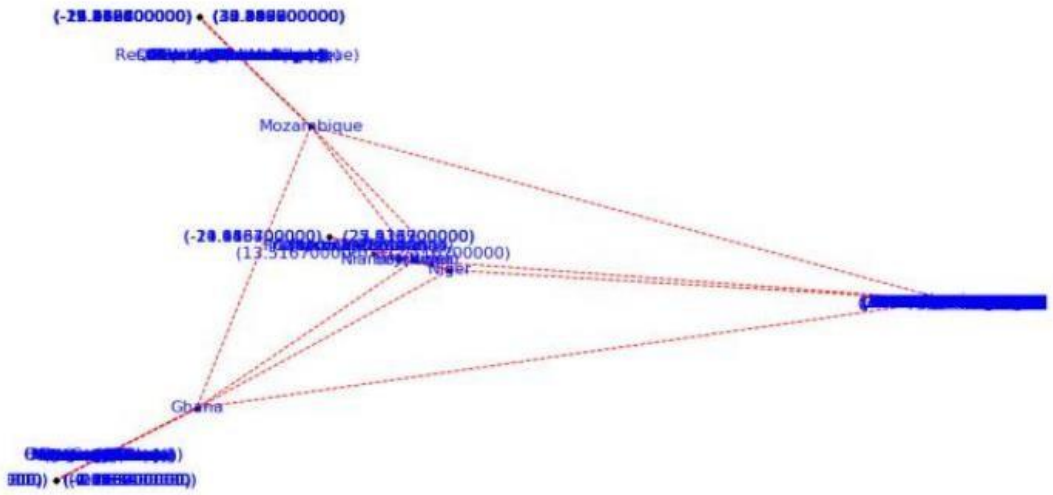
plt.savefig(sFileName,dpi=600)

plt.show()

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

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

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



import sys

import os

import pandas as pd from folium.plugins

import FastMarkerCluster, HeatMap from folium

import Marker, Map import webbrowser
##

Base='C:/VKHCG'

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

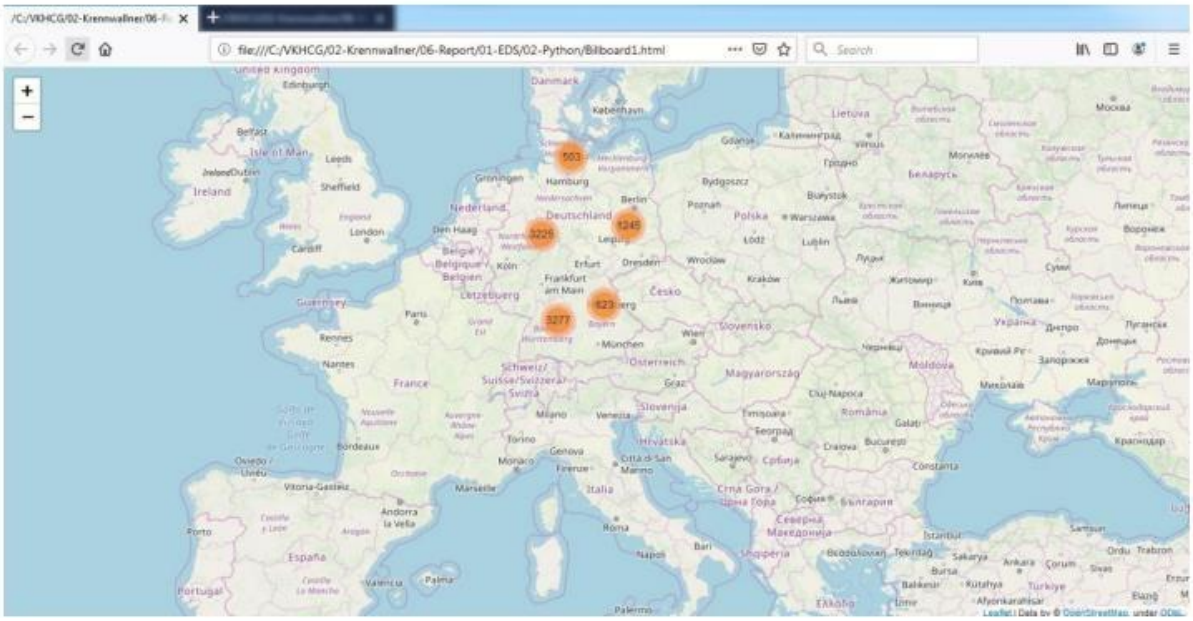
print('Working Base :',Base, ' using ', sys.platform)

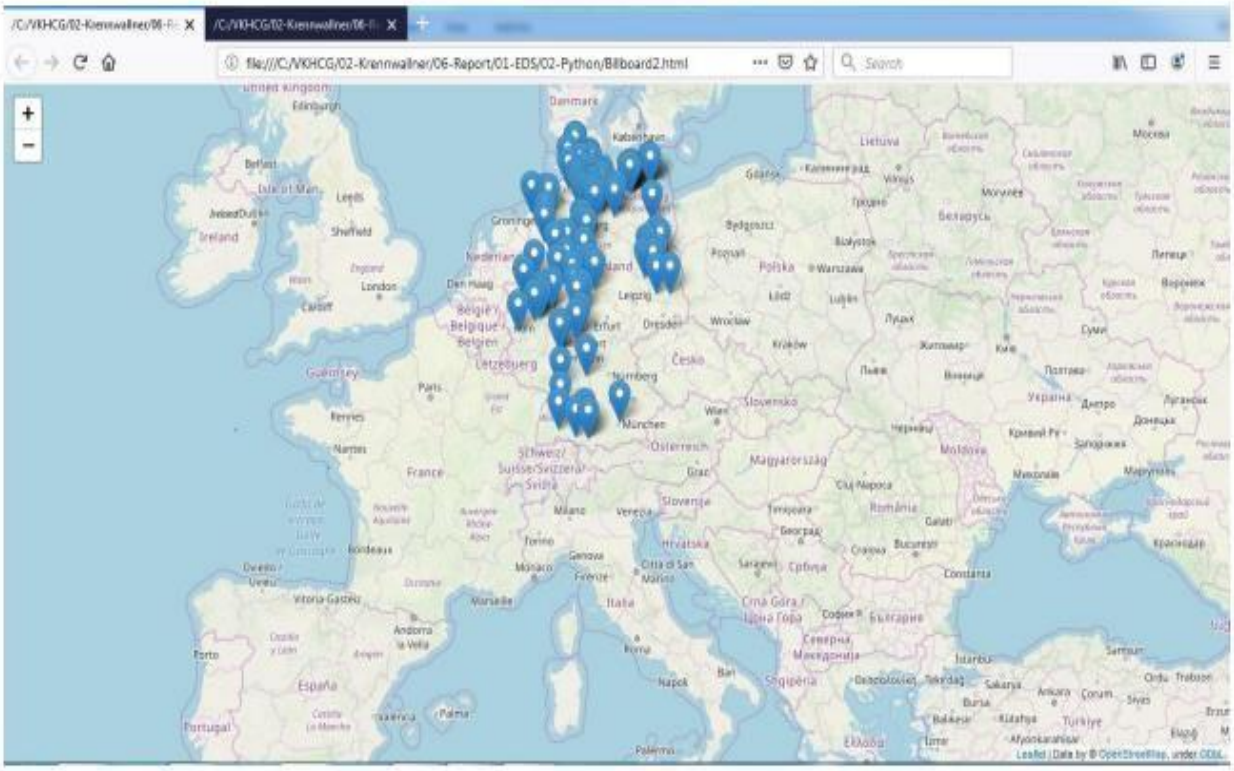
print('####################################')
##
sFileName=Base+'/02-Krennwallner/01-Retrieve/01-EDS/02-
Python/Retrieve_DE_Billboard_Locations.csv' df =
pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1") df.fillna(value=0,
inplace=True)

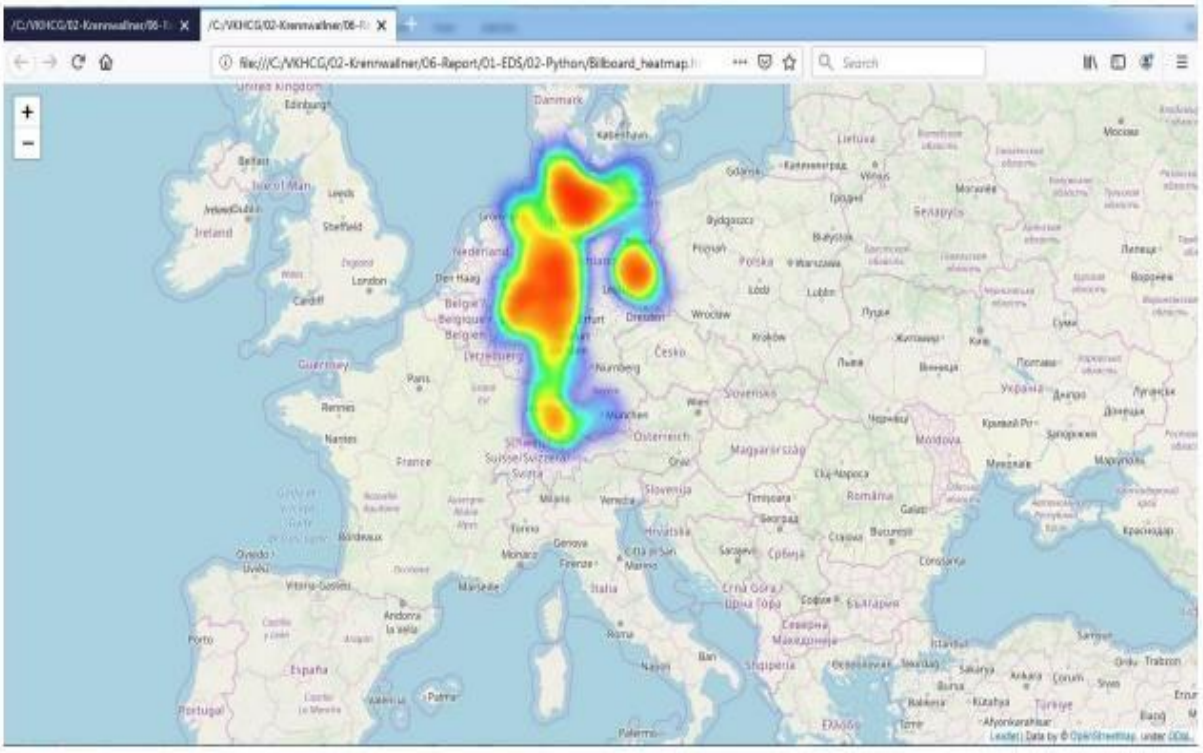
```
print(df.shape)
########################################################################## t=0 for i in
range(df.shape[0]): try:
sLongitude=df["Longitude"][i]
sLongitude=float(sLongitude) except Exception:
sLongitude=float(0.0) try:
sLatitude=df["Latitude"][i]
sLatitude=float(sLatitude)
except Exception:
sLatitude=float(0.0) try: s
Description=df["Place_Name"][i] + ' (' + df["Country"][i]+')' except Exception:
sDescription='VKHCG'
if sLongitude != 0.0 and
sLatitude != 0.0:
DataClusterList=list([sLatitude, sLongitude])
DataPointList=list([sLatitude, sLongitude, sDescription])
t+=1
if t==1:
DataCluster=[DataClusterList]
DataPoint=[DataPointList]
else: DataCluster.append(DataClusterList)
DataPoint.append(DataPointList)
data=DataCluster pins=pd.DataFrame(DataPoint)
pins.columns = [ 'Latitude','Longitude','Description']
########################################################################
stops_map1 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
marker_cluster = FastMarkerCluster(data).add_to(stops_map1)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard1.html'
stops_map1.save(sFileNameHtml) webbrowser.open('file://' +
os.path.realpath(sFileNameHtml))
########################################################################
stops_map2 = Map(location=[48.1459806, 11.4985484], zoom_start=5) for name, row in
pins.iloc[:100].iterrows():
Marker([row["Latitude"],row["Longitude"]],
popup=row["Description"]).add_to(stops_map2)
```

```
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html'
stops_map2.save(sFileNameHtml)

webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
##########################################################################

stops_heatmap = Map(location=[48.1459806, 11.4985484], zoom_start=5)
stops_heatmap.add_child(HeatMap([[row["Latitude"],

row["Longitude"]] for name, row in pins.iloc[:100].iterrows()])) sFileNameHtml=Base+'/02-
Krennwallner/06-Report/01-EDS/02-Python/Billboard_heatmap.html'
stops_heatmap.save(sFileNameHtml) webbrowser.open('file://' +
os.path.realpath(sFileNameHtml))
##########################################################################

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

Output:



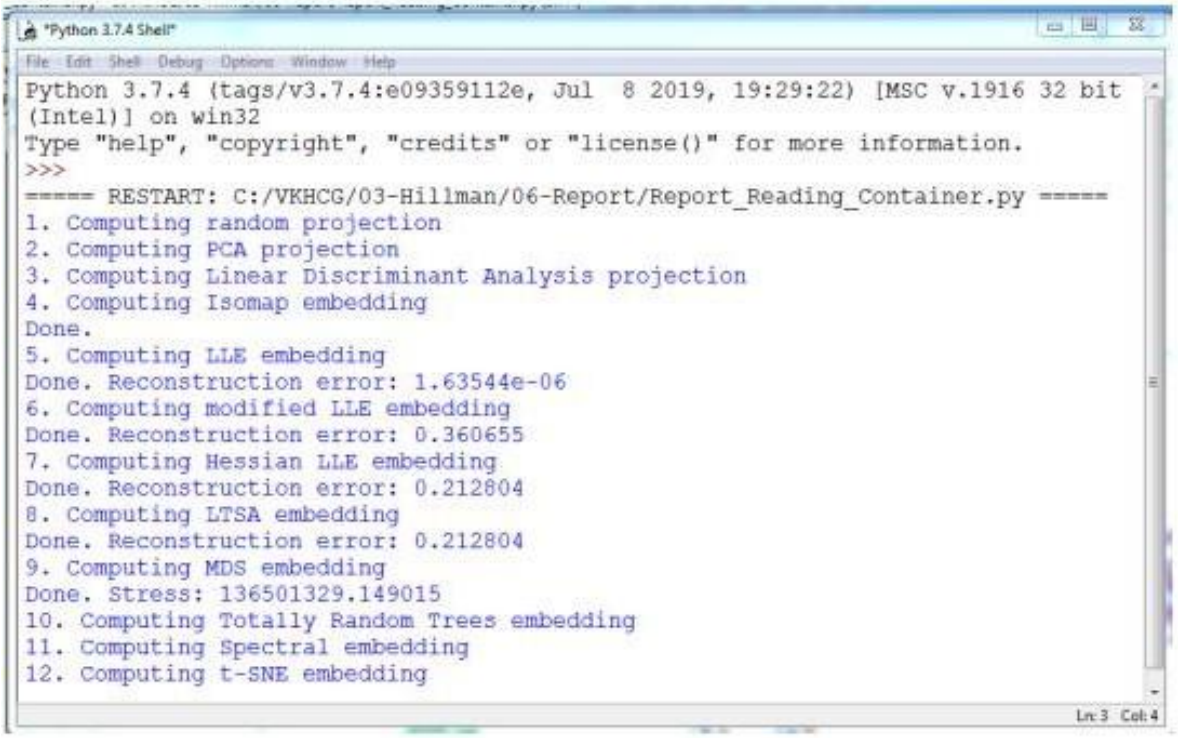

file:///C:/VKHCG/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html
Search
Leaflet | Data by © OpenStreetMap, under ODbL.


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

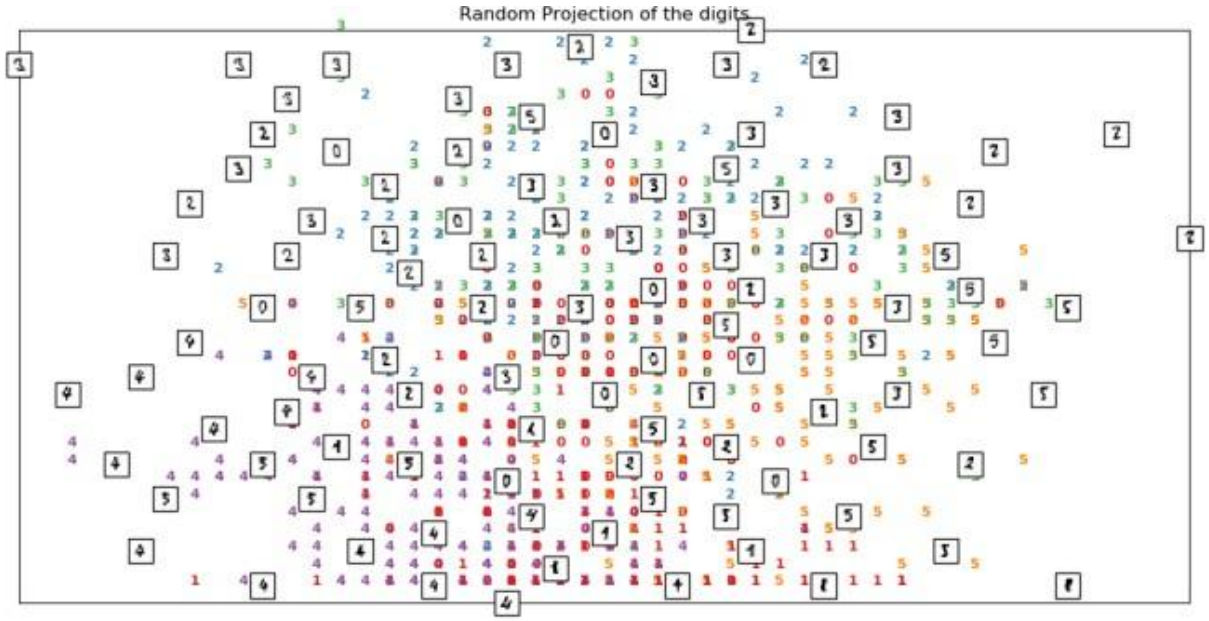
```
from time import time
import numpy as np
import matplotlib.pyplot as plt from matplotlib
import offsetbox from sklearn
import (manifold, datasets, decomposition, ensemble, discriminant_analysis,
random_projection) digits = datasets.load_digits(n_class=6)
X = digits.data y = digits.target n_samples,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

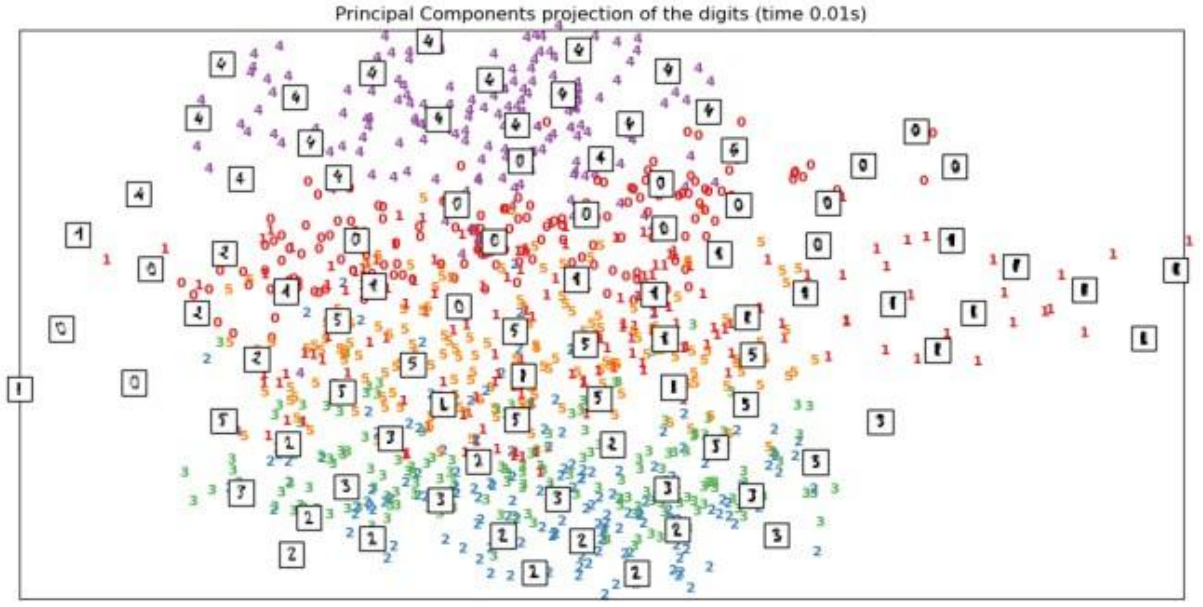
```
plt.figure(figsize=(10, 10))
plt.imshow(img, cmap=plt.cm.binary)
plt.xticks([]) plt.yticks([])
plt.title('A selection from the 64-dimensional digits dataset')
print("Computing random projection") rp =
random_projection.SparseRandomProjection(n_components=2, random_state=42)
X_projected = rp.fit_transform(X)
plot_embedding(X_projected, "Random Projection of the digits")
print("Computing PCA projection")
t0 = time()
X_pca = decomposition.TruncatedSVD(n_components=2).fit_transform(X)
plot_embedding(X_pca,"Principal Components projection of the digits (time %.2fs)"
%(time() - t0)) print("Computing Linear Discriminant Analysis projection")
 X2 = X.copy() X2.flat[::X.shape[1] + 1] += 0.01
# Make X invertible t0 = time() X_lda =
discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_transform(X2, y)
plot_embedding(X_lda,"Linear Discriminant projection of the digits (time %.2fs)" %(time() -
t0)) print("Computing Isomap embedding")
t0 = time()
X_iso = manifold.Isomap(n_neighbors, n_components=2).fit_transform(X)
print("Done.") plot_embedding(X_iso,"Isomap projection of the digits (time %.2fs)"
%(time() - t0)) print("Computing LLE embedding")
 clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,method='standard')
t0 = time() X_lle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
 plot_embedding(X_lle,"Locally Linear Embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing modified LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2, method='modified')
t0 = time()
 X_mlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_mlle,"Modified Locally Linear Embedding of the digits (time %.2fs)"
%(time() - t0))
print("Computing Hessian LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,method='hessian')
t0 = time()
```

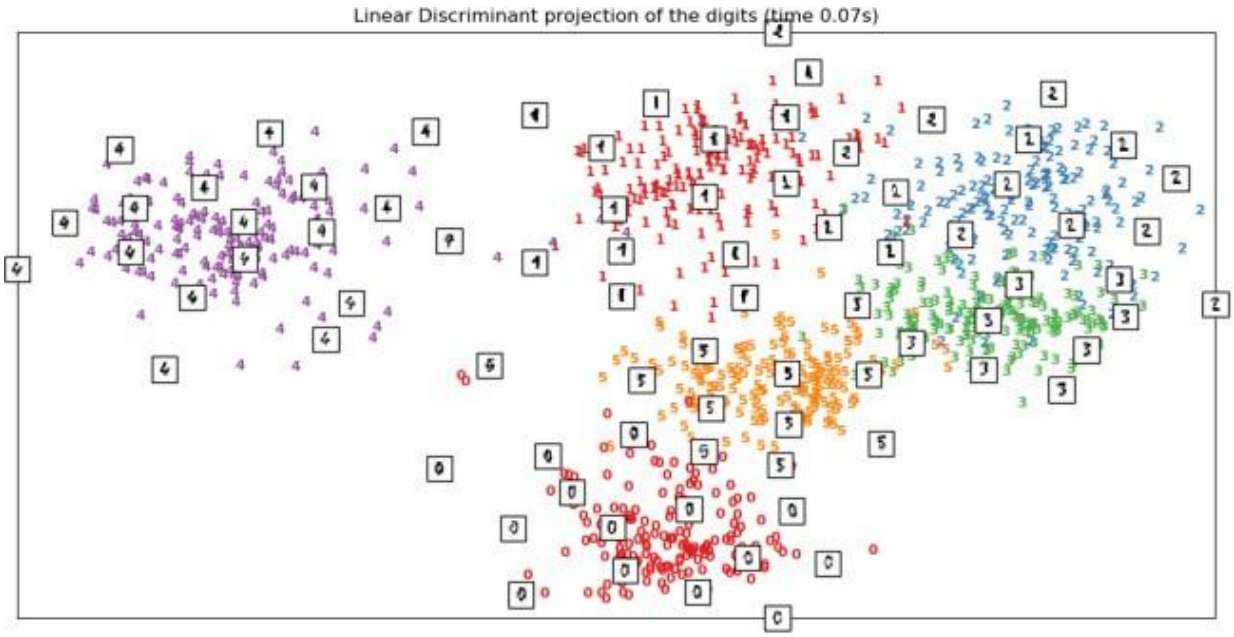
```
X_hlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_hlle,"Hessian Locally Linear Embedding of the digits (time %.2fs)"
%(time() - t0)) print("Computing LTSA embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,method='ltsa')
t0 = time()
X_ltsa = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_ltsa,"Local Tangent Space Alignment of the digits (time %.2fs)"
%(time() - t0)) print("Computing MDS embedding")
clf = manifold.MDS(n_components=2,
n_init=1, max_iter=100)
t0 = time()
 X_mds = clf.fit_transform(X)
 print("Done. Stress: %f" % clf.stress_)
plot_embedding(X_mds,"MDS embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing Totally Random Trees embedding")
hasher = ensemble.RandomTreesEmbedding(n_estimators=200, random_state=0,
max_depth=5) t0 = time() X_transformed = hasher.fit_transform(X)
pca = decomposition.TruncatedSVD(n_components=2)
X_reduced = pca.fit_transform(X_transformed)
plot_embedding(X_reduced,"Random forest embedding of the digits (time %.2fs)" %(time()
- t0)) print("Computing Spectral embedding")
embedder = manifold.SpectralEmbedding(n_components=2, random_state=
eigen_solver="arpack")
t0 = time()
X_se = embedder.fit_transform(X)
plot_embedding(X_se,"Spectral embedding of the digits (time %.2fs)" %(time() - t0))
print("Computing t-SNE embedding")
tsne = manifold.TSNE(n_components=2,
init='pca', random_state=0)
t0 = time()
X_tsne = tsne.fit_transform(X)
plot_embedding(X_tsne,"t-SNE embedding of the digits (time %.2fs)" %(time() - t0))
plt.show()
```

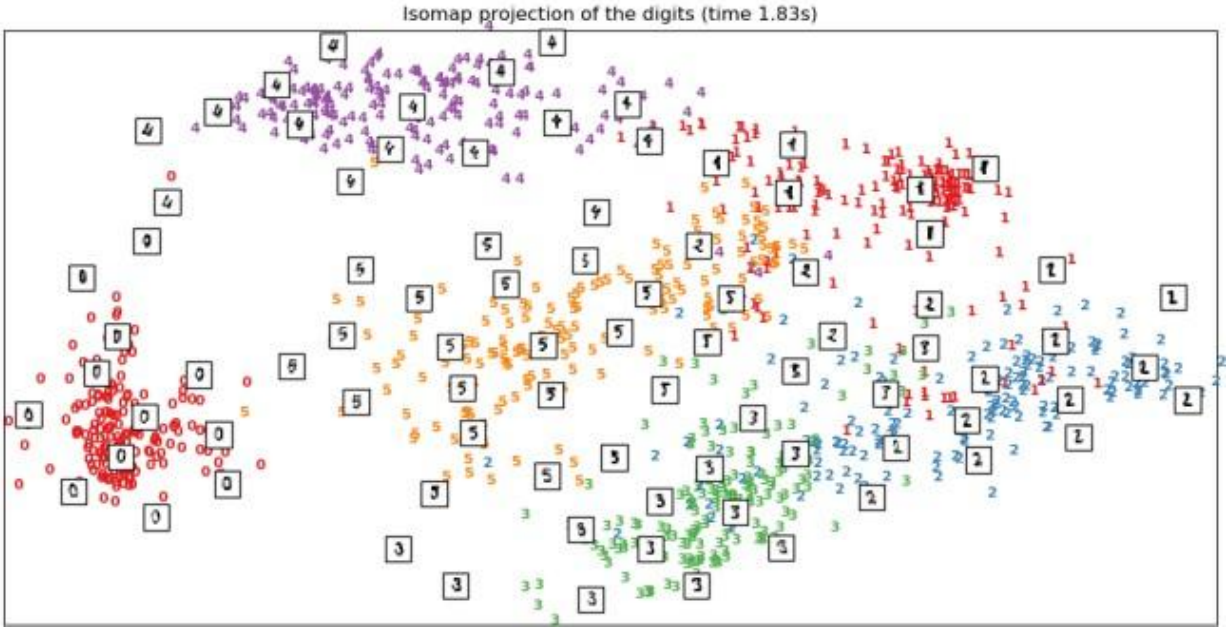


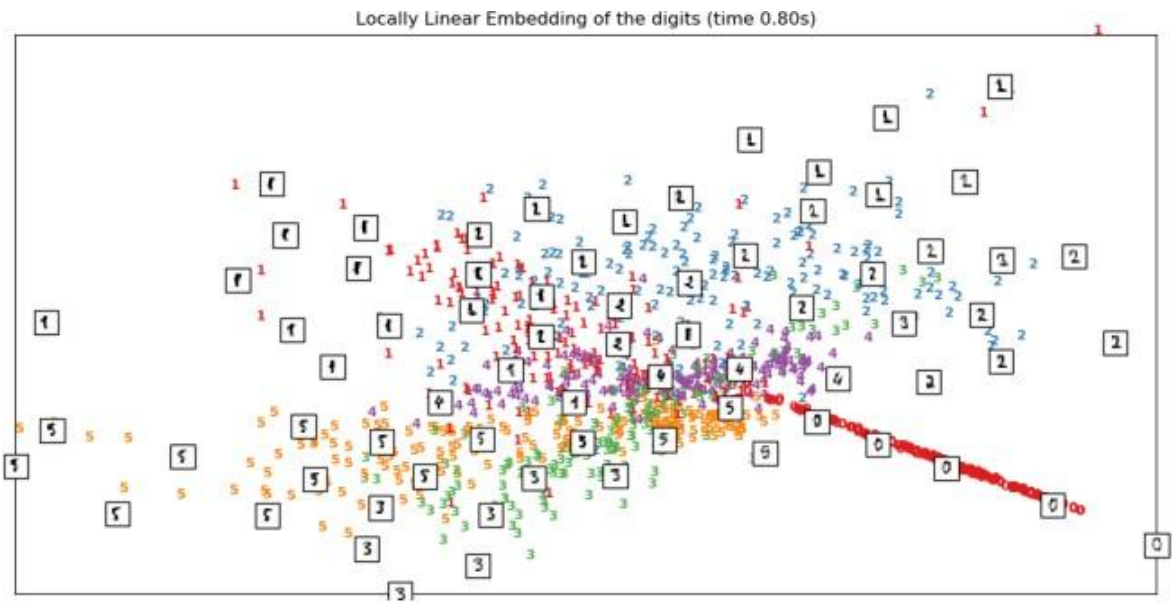
A selection from the 64-dimensional digits dataset

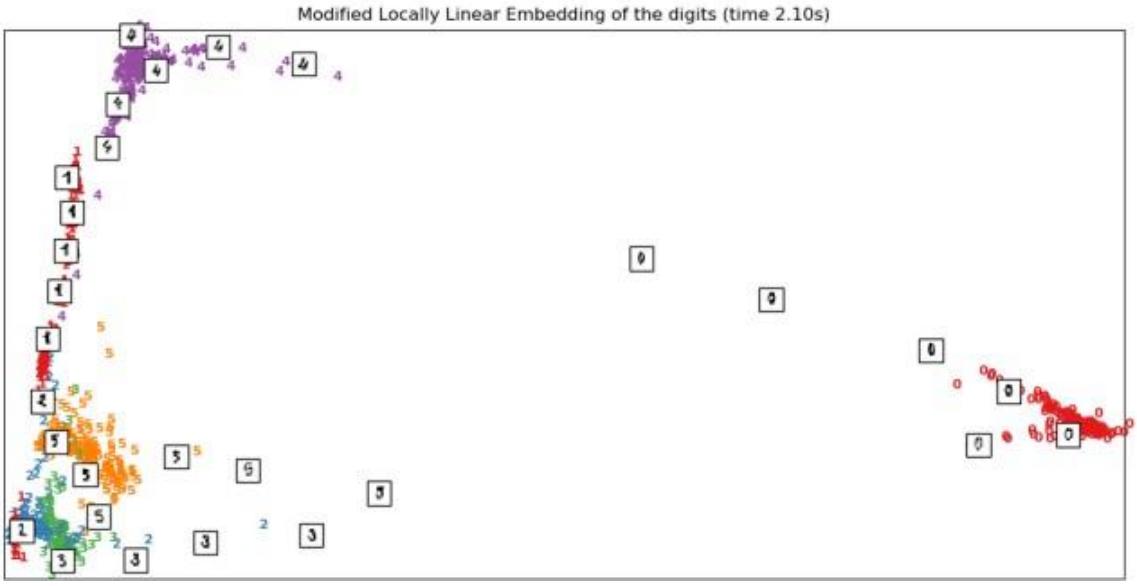

Random Projection of the digits

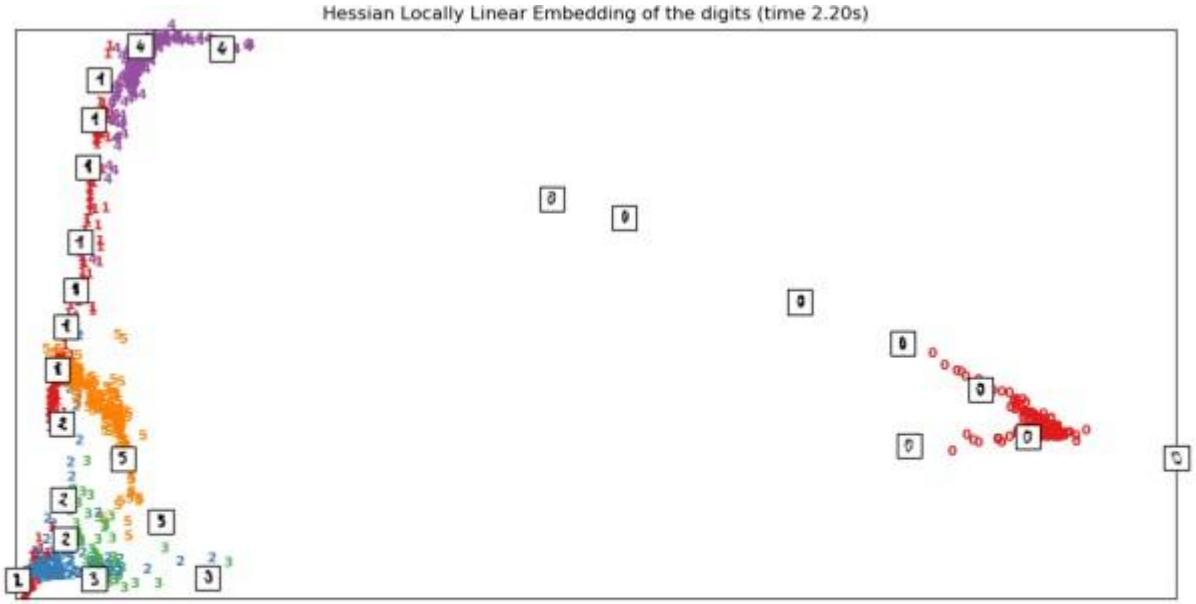

Principal Components projection of the digits (time 0.01s)

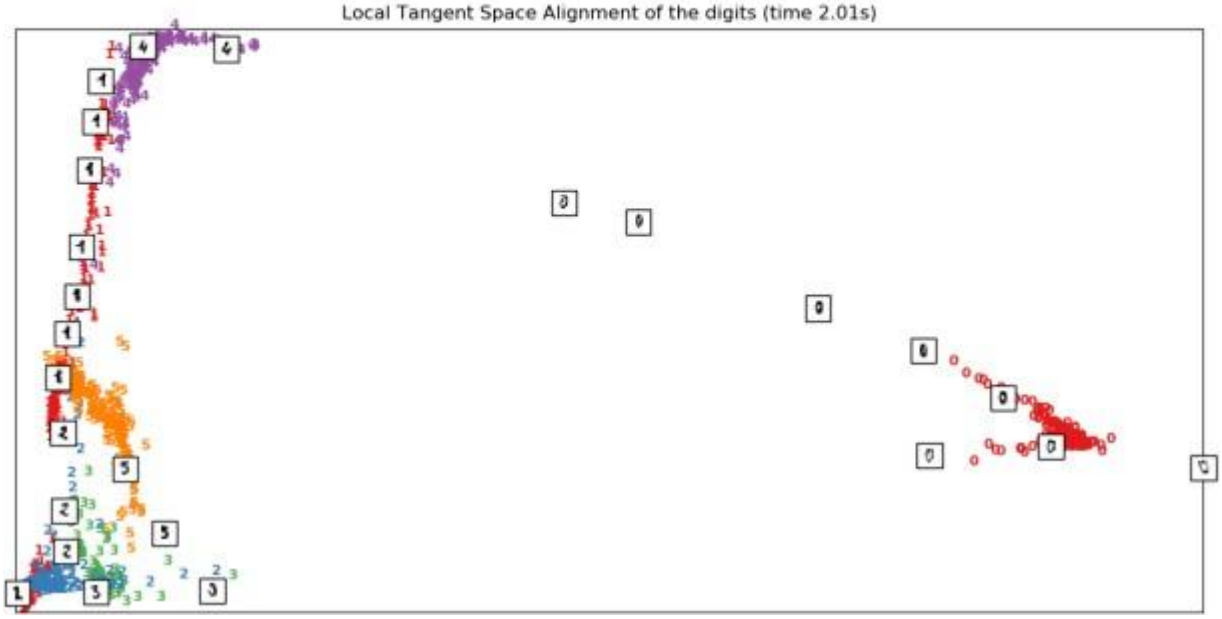

Linear Discriminant projection of the digits (time 0.07s)

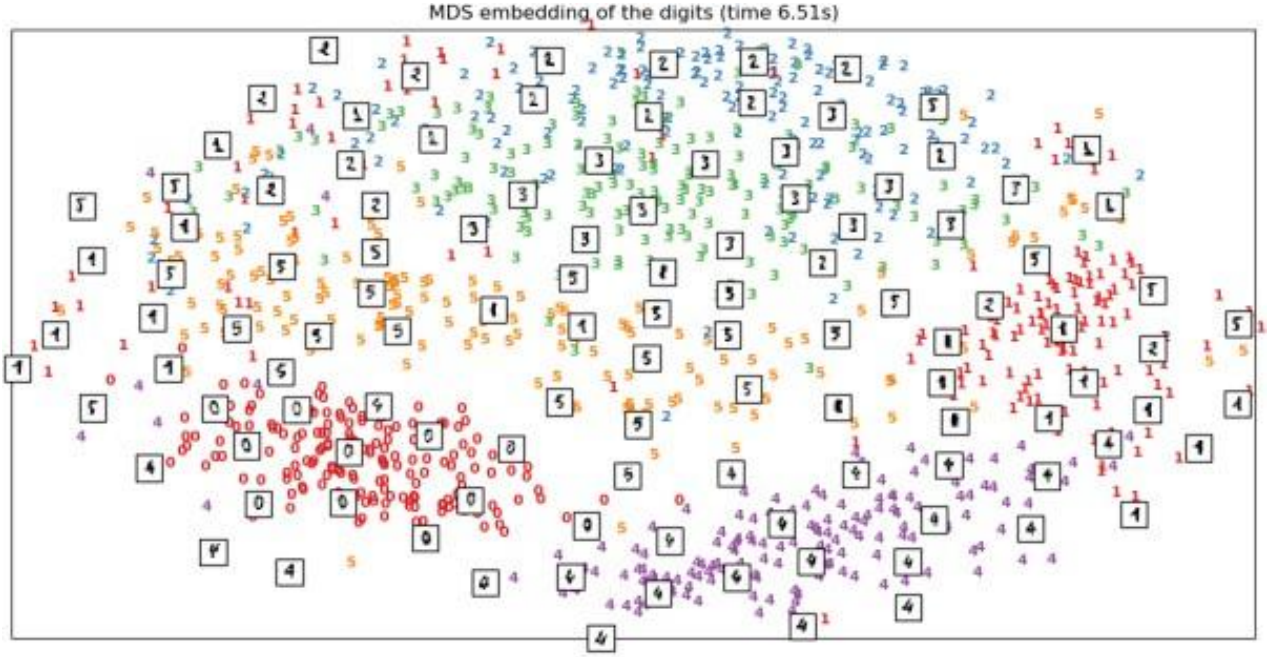

Isomap projection of the digits (time 1.83s)

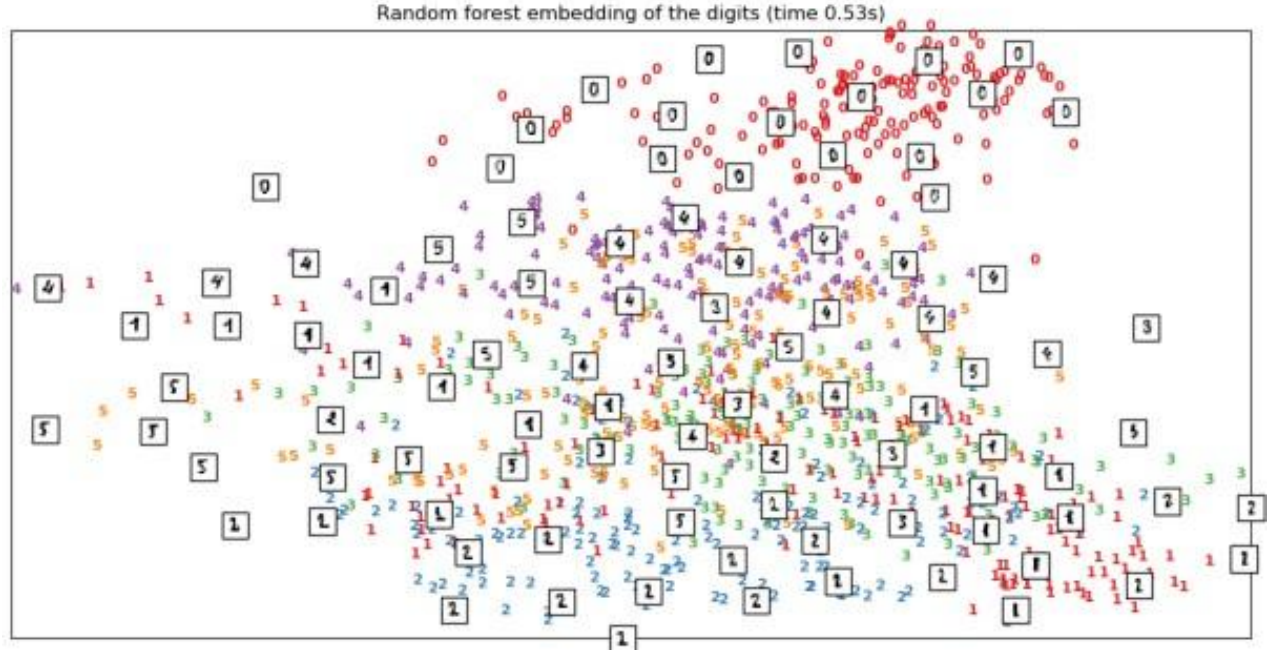

Locally Linear Embedding of the digits (time 0.80s)

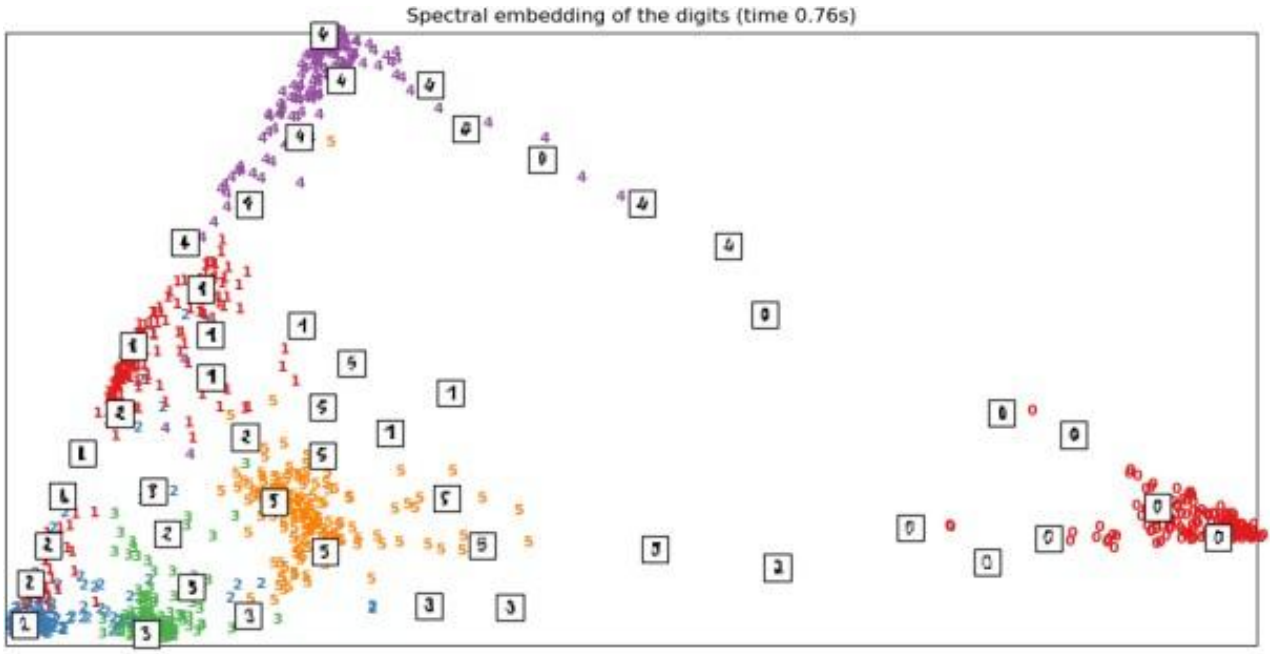

Modified Locally Linear Embedding of the digits (time 2.10s)

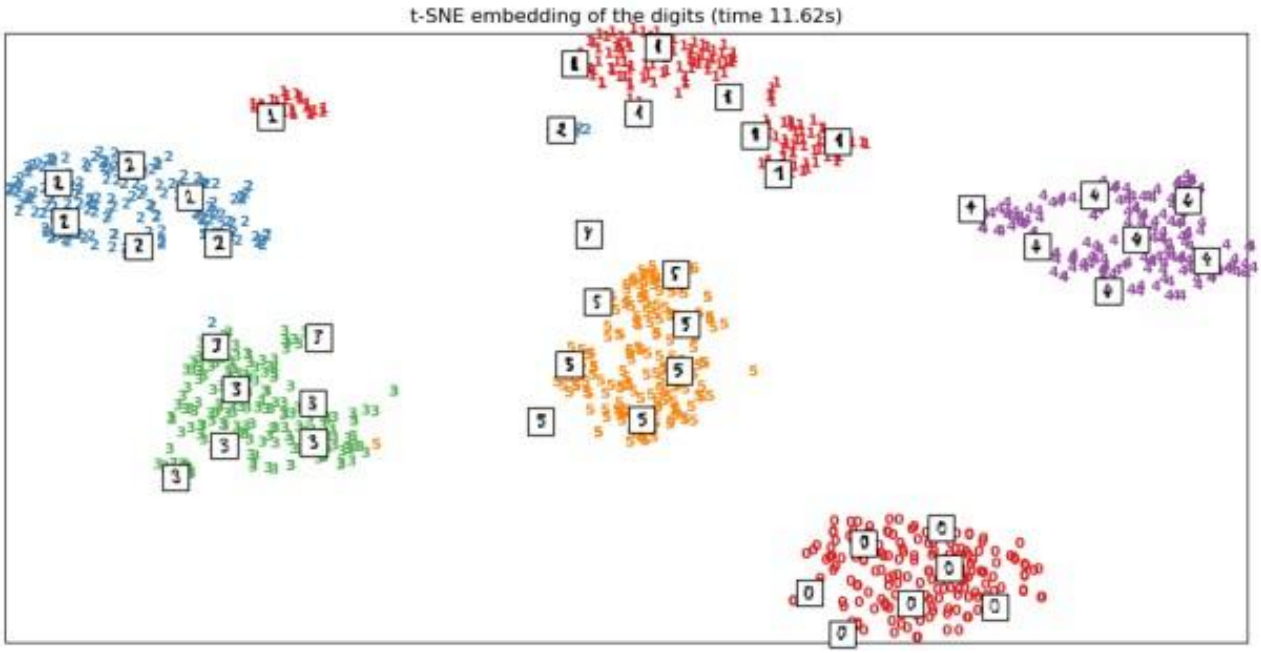

Hessian Locally Linear Embedding of the digits (time 2.20s)


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


MDS embedding of the digits (time 6.51s)


Random forest embedding of the digits (time 0.53s)






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

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

```
sCompanyFile \ FROM BalanceSheets \ GROUP BY \ Year, \ Quarter, \ Country, \ Company
\ HAVING Year is not null \ ;" sSQL=re.sub("\s\s+", " ", sSQL)

sDatesRaw=pd.read_sql_query(sSQL, conn)

print(sDatesRaw.shape)

sDates=sDatesRaw.head(5)
########################################################################

## Loop Dates
######################################################################## for i in
range(sDates.shape[0]): sFileName=Base + '/' + Company + '/' +

sInputTemplateName wb = load_workbook(sFileName)

ws=wb.get_sheet_by_name("Balance-Sheet")

sYear=sDates['sDate'][i]

sCompany=sDates['sCompanyName'][i]

sCompanyFile=sDates['sCompanyFile'][i]

sCompanyFile=re.sub("\s+", "", sCompanyFile)

ws['D3'] = sYear ws['D5'] =

sCompany sFields = pd.DataFrame( [ ['Cash','D16', 1],

['Accounts_Receivable','D17', 1],

['Doubtful_Accounts','D18', 1],

['Inventory','D19', 1],

['Temporary_Investment','D20', 1],

['Prepaid_Expenses','D21', 1],

['Long_Term_Investments','D24', 1],

['Land','D25', 1],

['Buildings','D26', 1],

['Depreciation_Buildings','D27', -1],

['Plant_Equipment','D28', 1],

['Depreciation_Plant_Equipment','D29', -1],

['Furniture_Fixtures','D30', 1],

['Depreciation_Furniture_Fixtures','D31', -1],

['Accounts_Payable','H16', 1],

['Short_Term_Notes','H17', 1],

['Current_Long_Term_Notes','H18', 1],

['Interest_Payable','H19', 1],

['Taxes_Payable','H20', 1],
```

```
['Accrued_Payroll','H21', 1],
['Mortgage','H24', 1],
['Other_Long_Term_Liabilities','H25', 1],
['Capital_Stock','H30', 1] ] )
nYear=str(int(sDates['Year'][i]))
nQuarter=str(int(sDates['Quarter'][i]))
sCountry=str(sDates['Country'][i])
sCompany=str(sDates['Company'][i])
sFileName=Base + '/' + Company + '/' +
sOutputFileName + \ '-' + sCompanyFile + '.xlsx'
print(sFileName)
for j in range(sFields.shape[0]):
sSumField=sFields[0][j]
sCellField=sFields[1][j]
nSumSign=sFields[2][j]
sSQL="SELECT \ Year, \ Quarter, \ Country, \ Company, \ SUM(" + sSumField + ") AS
nSumTotal \ FROM BalanceSheets \ GROUP BY \ Year, \ Quarter, \ Country, \ Company \
HAVING \ Year=" + nYear + " \ AND \ Quarter=" + nQuarter + " \ AND \ Country='" +
sCountry + "' \
AND \ Company='" + sCompany + "' \ ;"
sSQL=re.sub("\s\s+", " ", sSQL)
sSumRaw=pd.read_sql_query(sSQL, conn)
ws[sCellField] = sSumRaw["nSumTotal"][0] * nSumSign print('Set cell',sCellField,' to ',
sSumField,'Total') wb.save(sFileName)
```

Output:

You now have all the reports you need.

Check the Following files for generated reports in C:/VKHCG/04-Clark/06-Report/01-EDS/02-Python/

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

Graphics

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

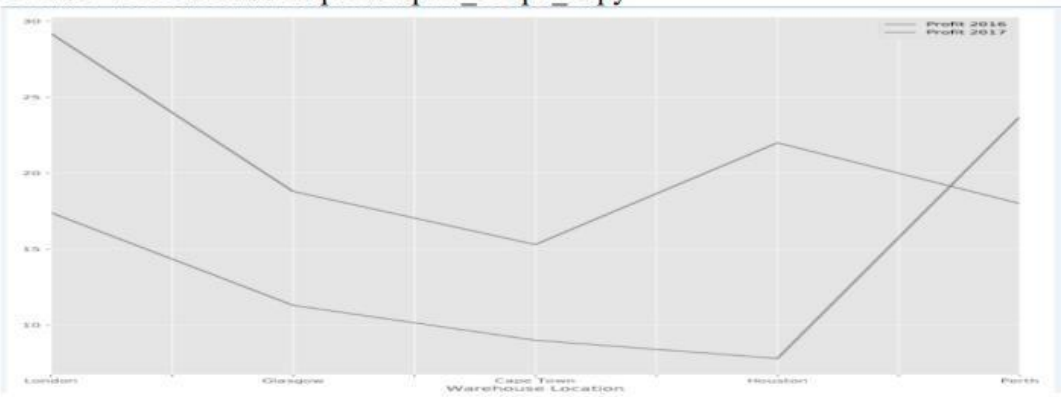
Pie Graph

Double Pie

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

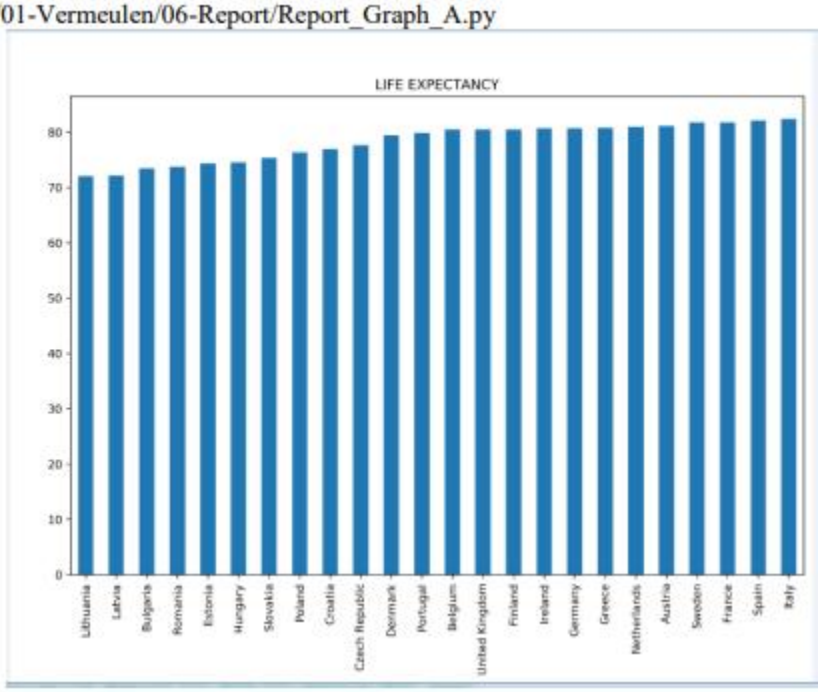
Line Graph

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



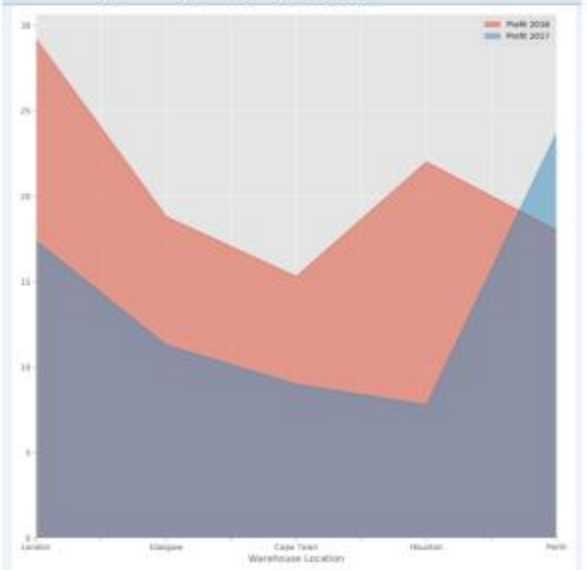
Bar Graph / Horizontal Bar Graph

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

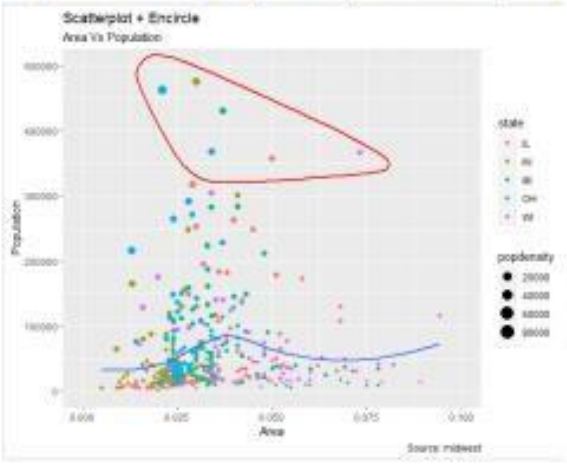


Area Graph

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

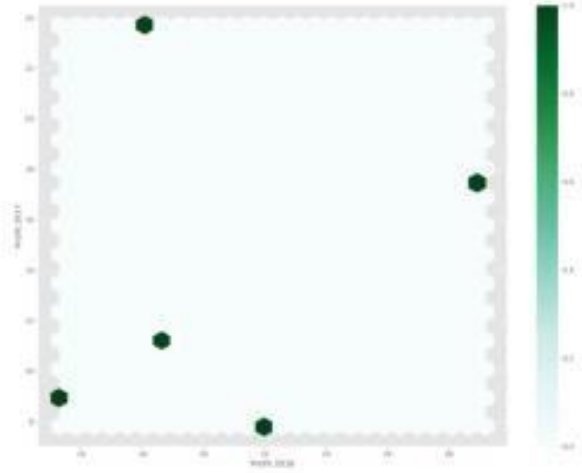


Scatter Graph : VKHCG/03-Hillman/06-Report/Report-Scatterplot-With-Encircling.r



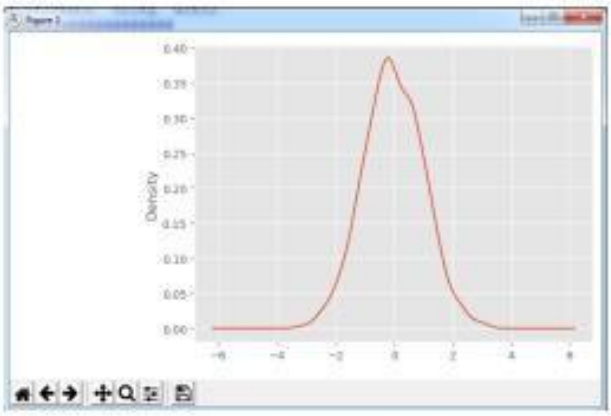
Hexbin:

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



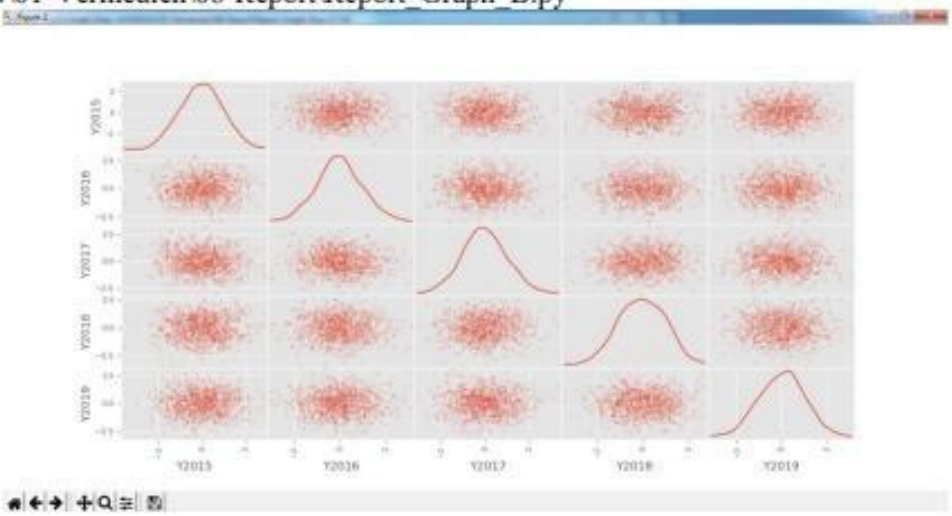
Kernel Density Estimation (KDE) Graph

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



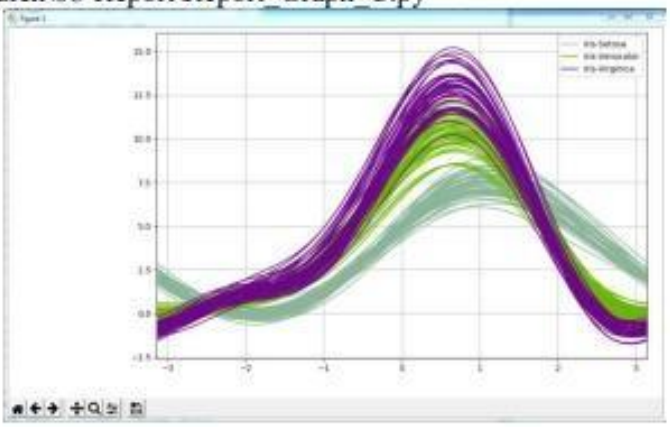
Scatter Matrix Graph

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



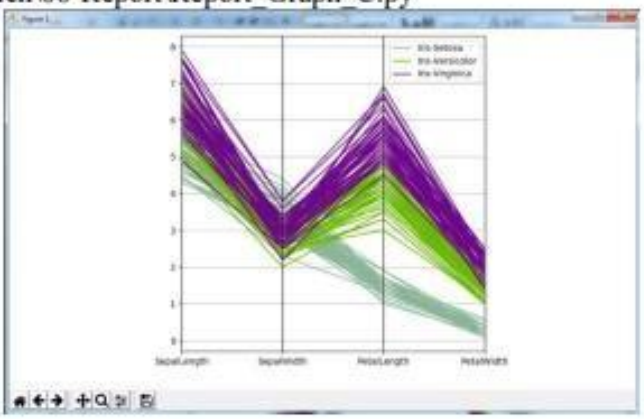
Andrews' Curves

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



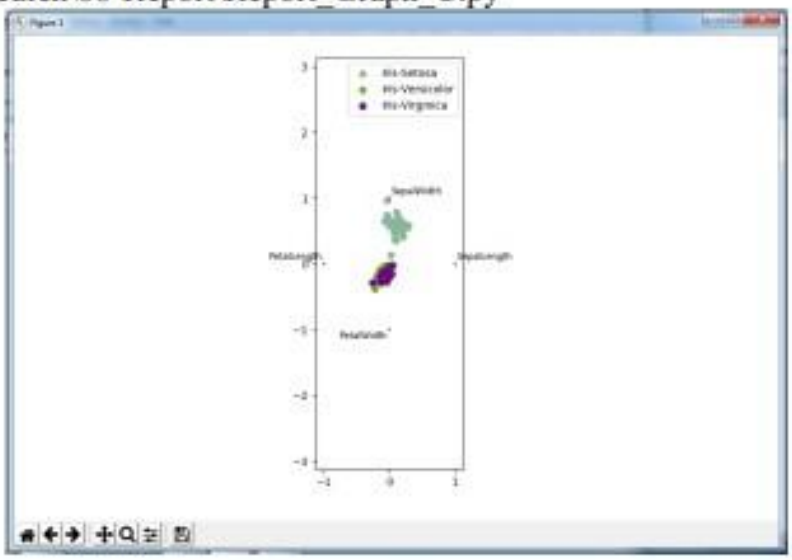
Parallel Coordinates

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



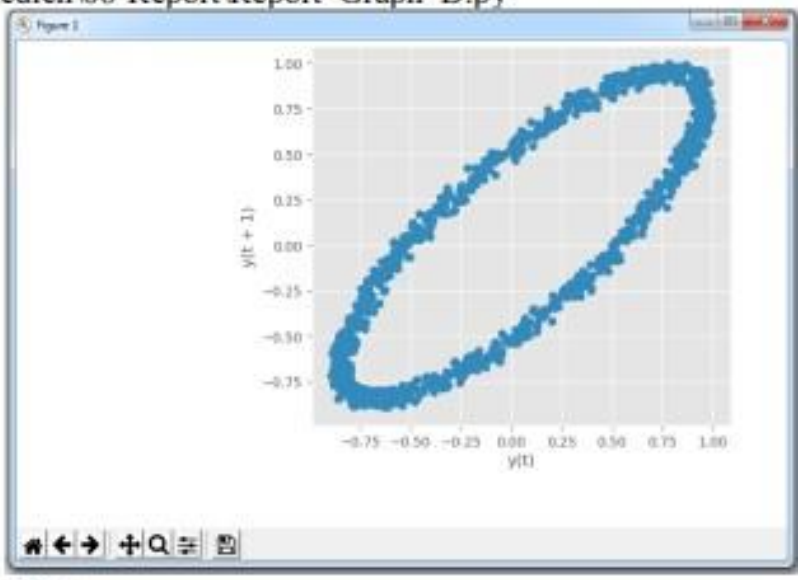
RADVIZ Method

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



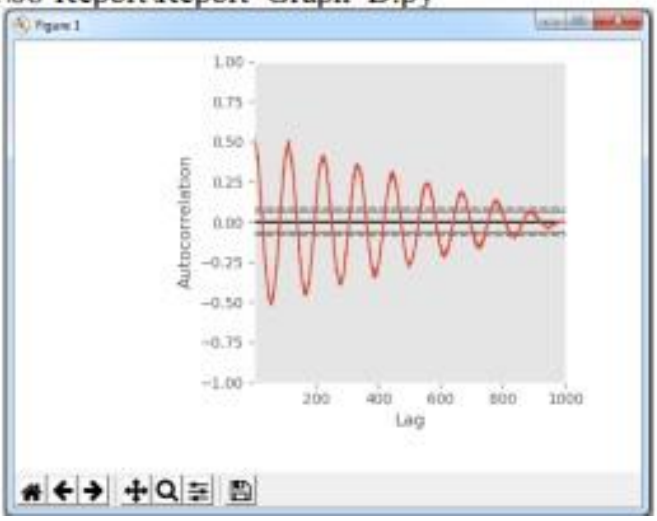
Lag Plot

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



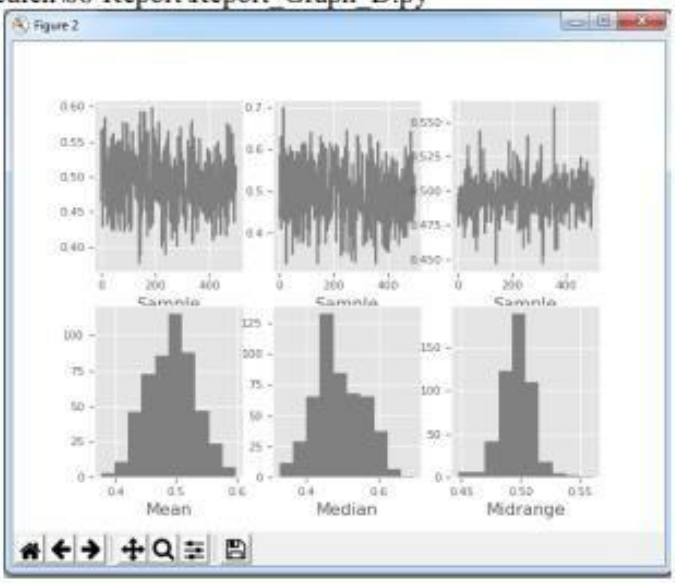
Autocorrelation Plot

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



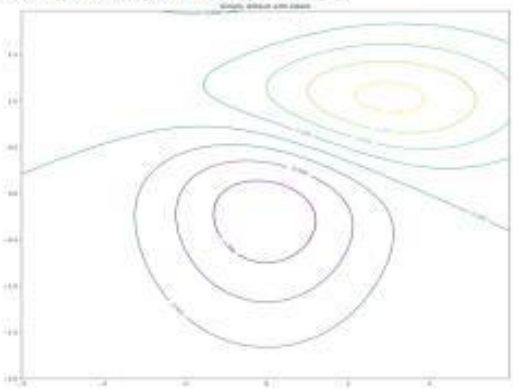
Bootstrap Plot

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



Contour Graphs

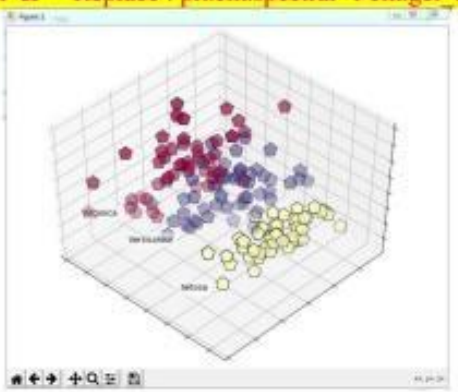
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_G.py



3D Graphs

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

(add →import matplotlib.cm as cm & Replace : plt.cm.spectral →cm.get_cmap("Spectral") at Line 44)

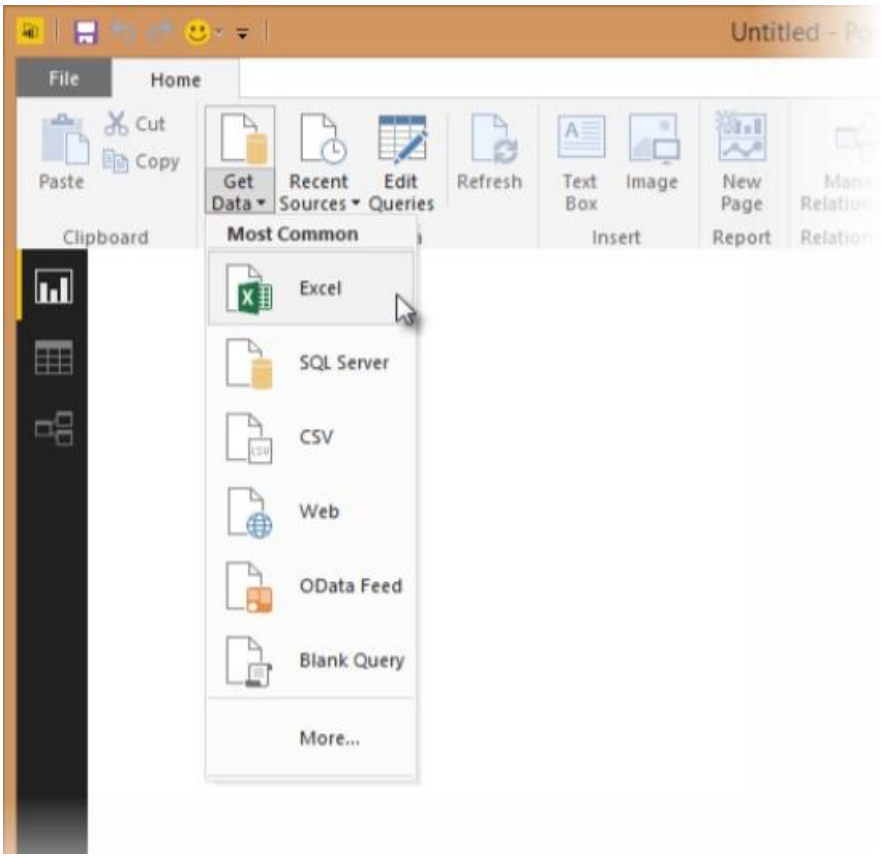


Practical 10

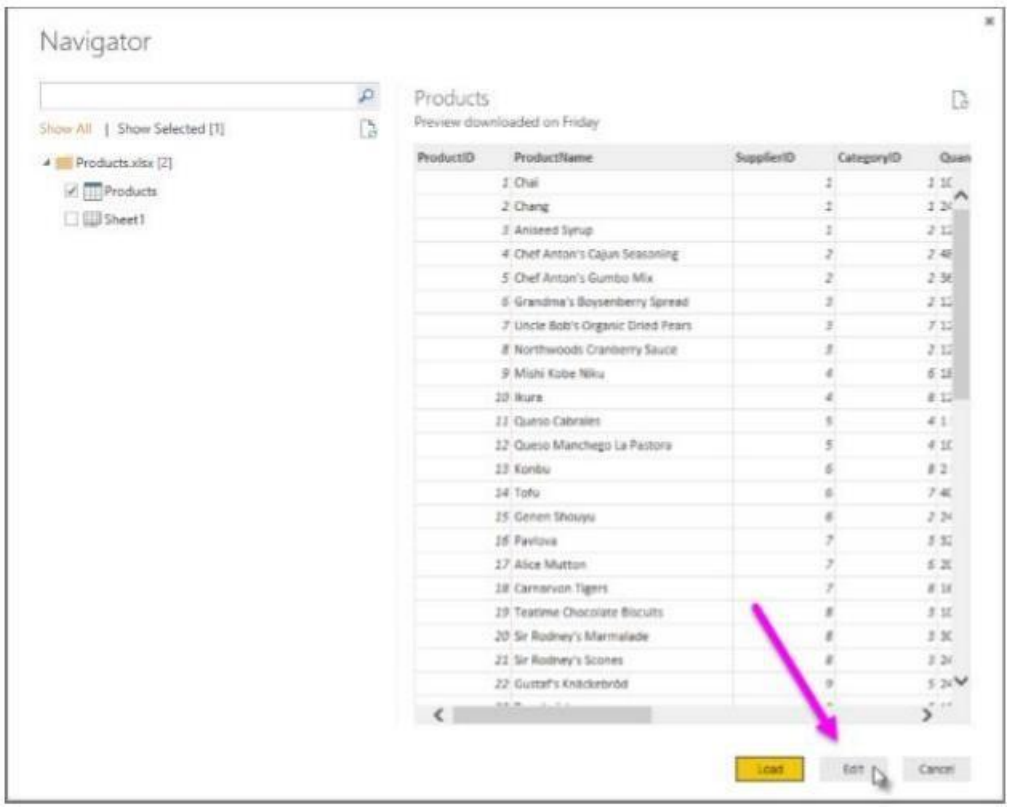
Importing Excel Data

1) Launch Power BI Desktop.

2) From the Home ribbon, select Get Data. Excel is one of the Most Common data connections, so you can select it directly from the Get Data menu



3) If you select the Get Data button directly, you can also select FIle > Excel and select Connect. 4) In the Open File dialog box, select the Products.xlsx file. 5) In the Navigator pane, select the Products table and then select Edit.



Importing Data from OData Feed In this task, you'll bring in order data. This step represents connecting to a sales system.

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/ 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. 5) In the Navigator pane, select the Orders table, and then select Edit

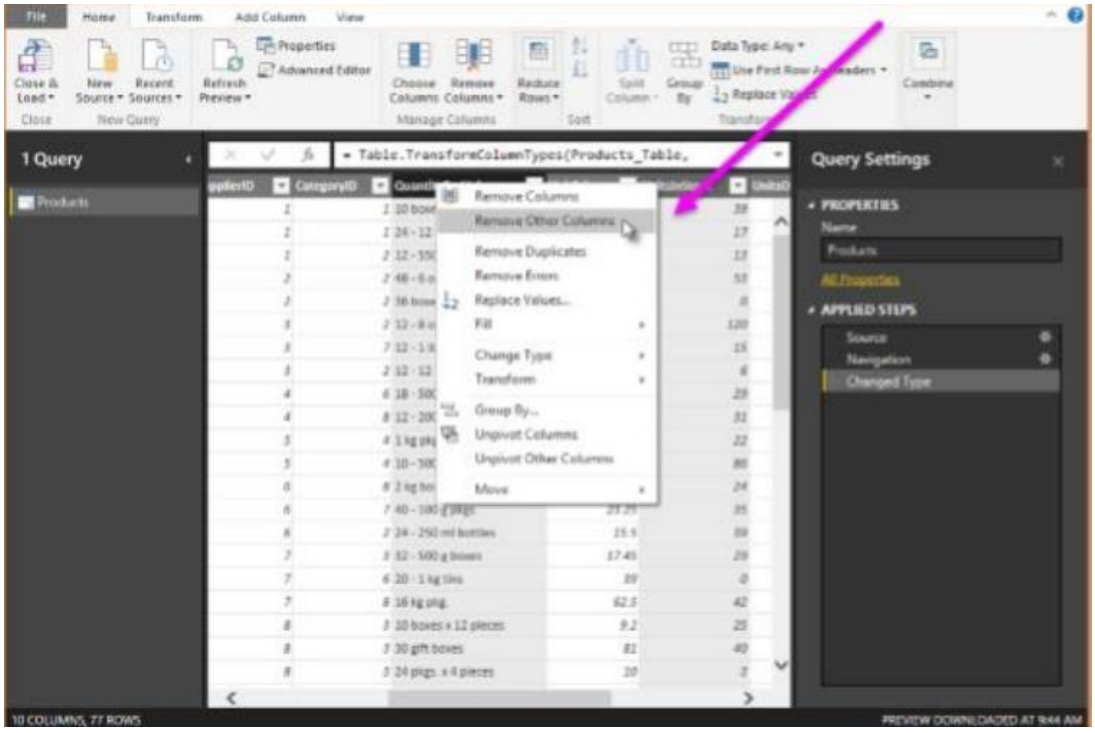
ETL Process in Power BI

1) Remove other columns to only display columns of interest

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

Power BI Desktop includes Query Editor, which is where you shape and transform your data connections. Query Editor opens automatically when you select **Edit** from Navigator. You can also open the Query Editor by selecting Edit Queries from the Home ribbon in Power BI Desktop. The following steps are performed in Query Editor.

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



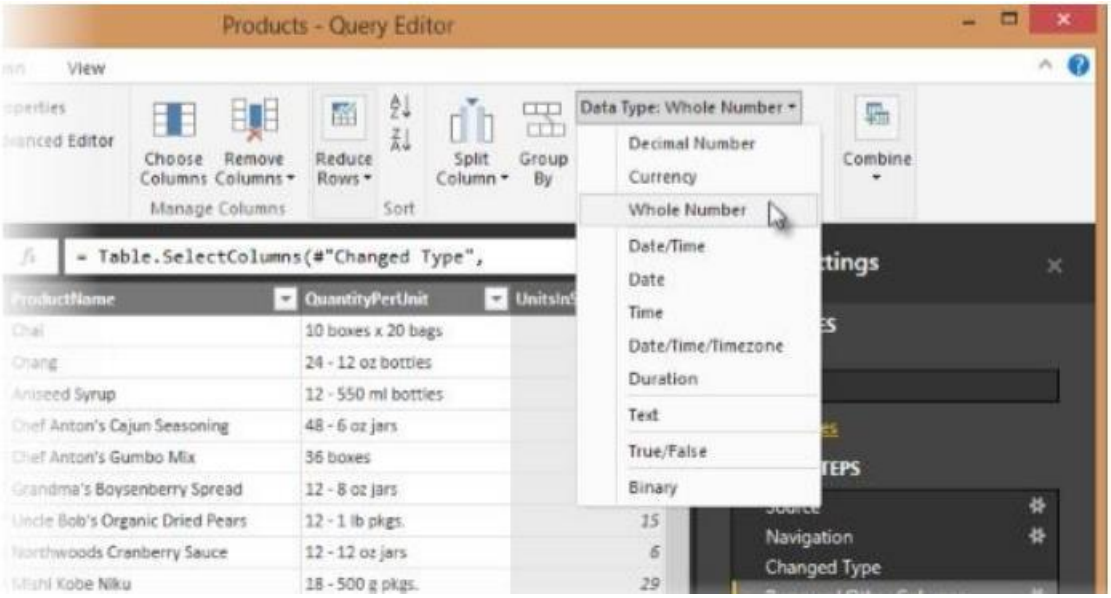
3. Change the data type of the UnitsInStock column

When Query Editor connects to data, it reviews each field and to determine the best data type For the Excel workbook, products in stock will always be a whole number, so in this step you confirm the **UnitsInStock** column's datatype is Whole Number.

1. Select the **UnitsInStock** column.

2. Select the **Data Type drop-down button** in the **Home ribbon**.

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



3. Expand the Order_Details table

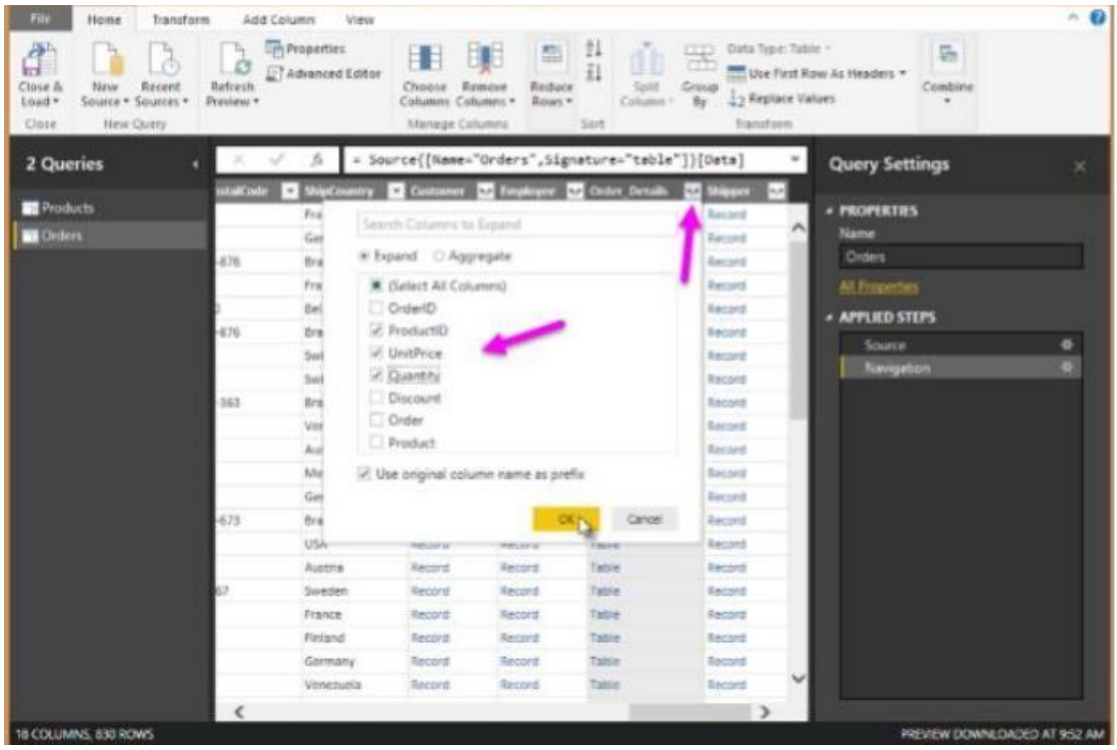
The Orders table contains a reference to a Details table, which contains the individual products that were included in each Order. When you connect to data sources with multiples tables (such as a relational database) you can use these references to build up your query

In this step, you expand the **Order_Details** table that is related to the Orders table, to combine the **ProductID**, **UnitPrice**, and **Quantity** columns from **Order_Details** into the **Orders table**. This is a representation of the data in these tables:

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

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

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

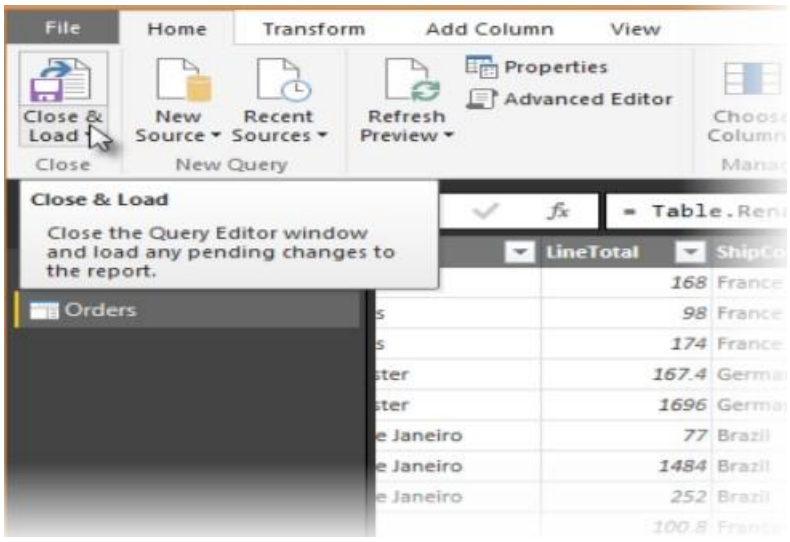


4. Calculate the line total for each Order_Details row

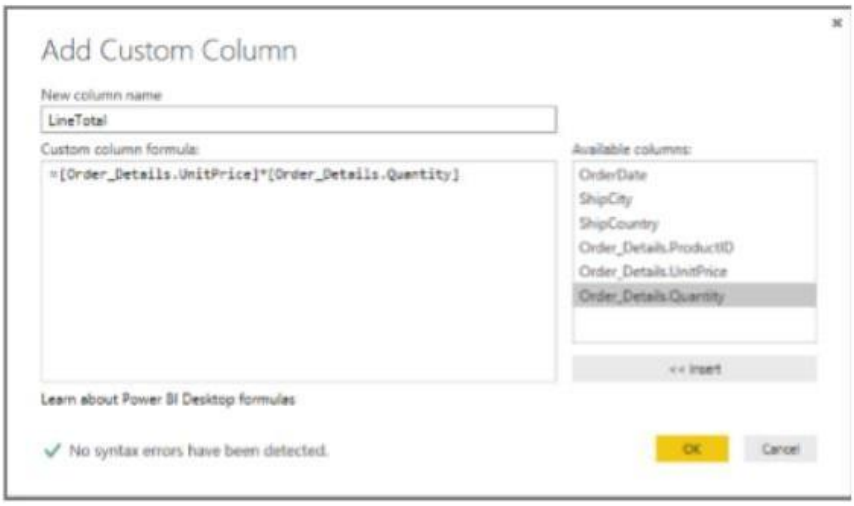
Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order_Details row.

Calculate the line total for each Order_Details row:

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



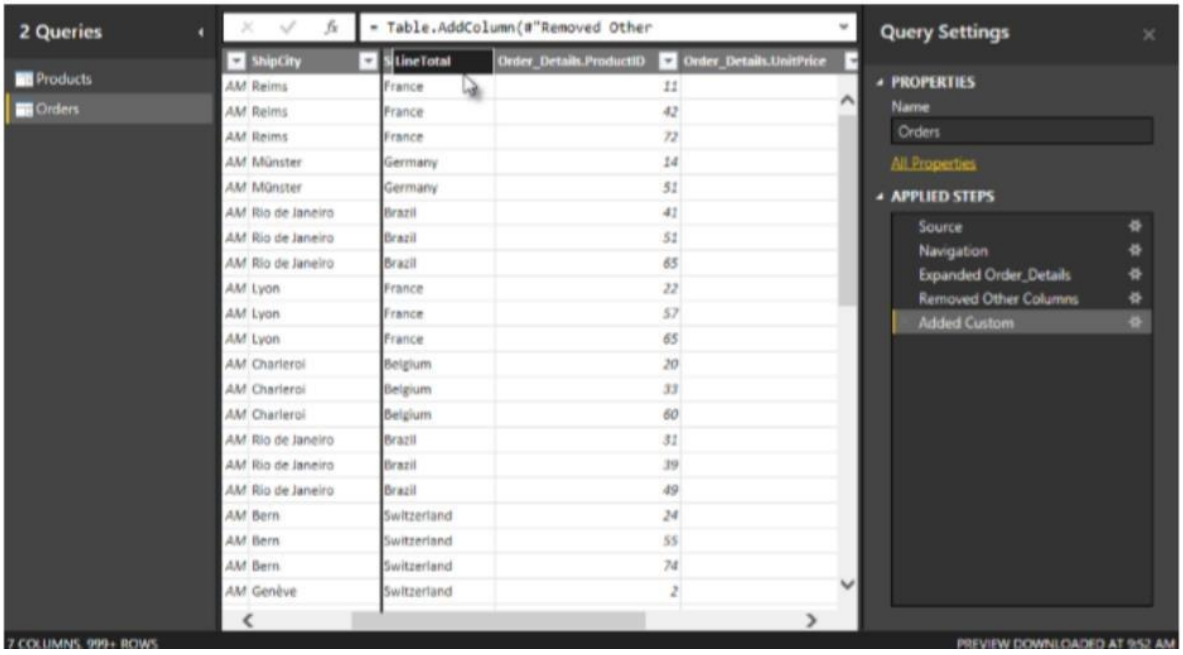
2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order_Details.UnitPrice] * [Order_Details.Quantity].
3. In the New column name textbox, enter LineTotal.
4. Click OK.



5. Rename and reorder columns in the query

In this step you finish making the model easy to work with when creating reports, by renaming the final columns and changing their order.

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.

6. Combine the Products and Total Sales queries

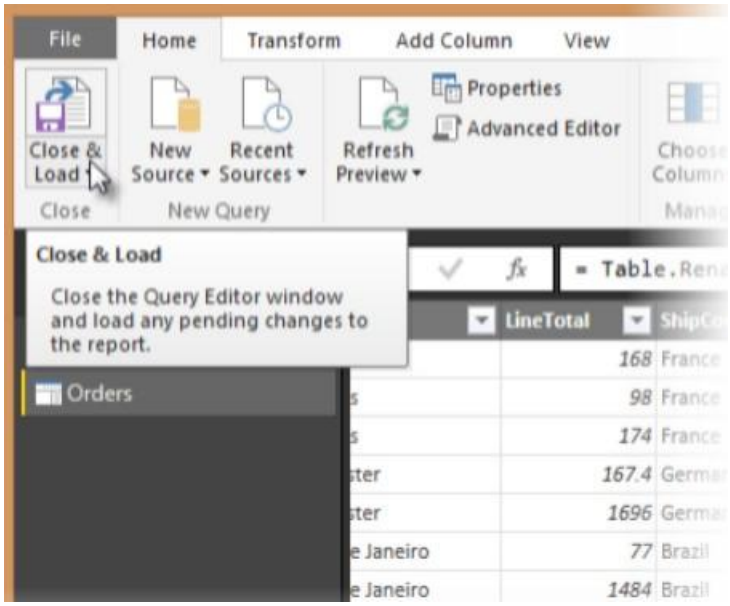
Power BI Desktop does not require you to combine queries to report on them. Instead, you can create Relationships between datasets. These relationships can be created on any column that is common to your datasets

we have Orders and Products data that share a common 'ProductID' field, so we need to ensure there's a relationship between them in the model we're using with Power BI Desktop. Simply specify in Power BI Desktop that the columns from each table are related (i.e. columns that have the same values). Power BI Desktop works out the direction and cardinality of the relationship for you. In some cases, it will even detect the relationships automatically.

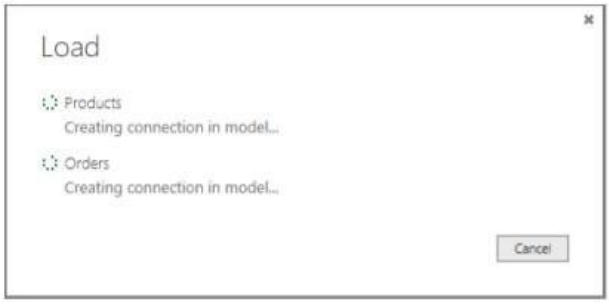
In this task, you confirm that a relationship is established in Power BI Desktop between the Products and Total Sales queries

Step 1: Confirm the relationship between Products and Total Sales

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

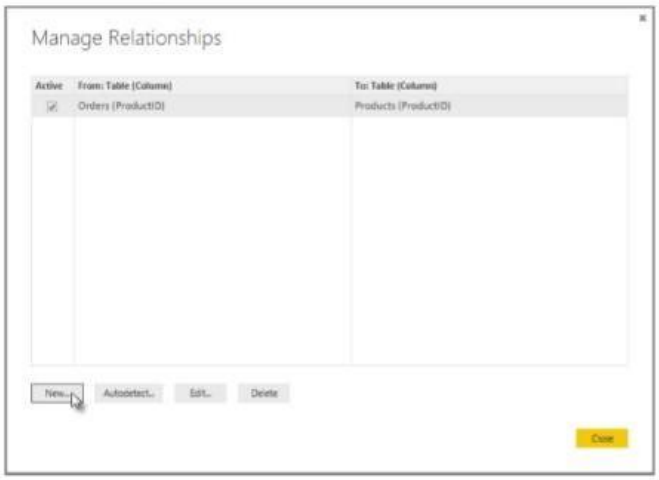


2. Power BI Desktop loads the data from the two queries.

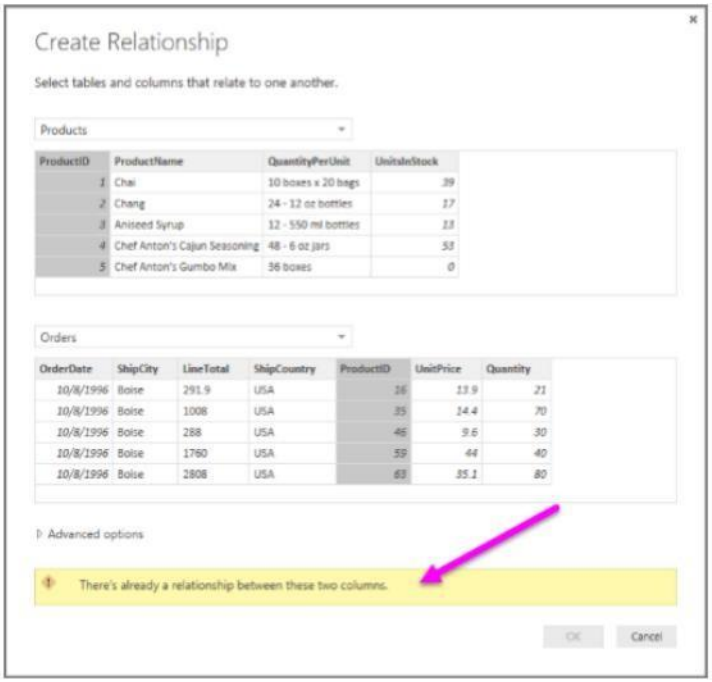


3. Once the data is loaded, select the Manage Relationships button Home ribbon.

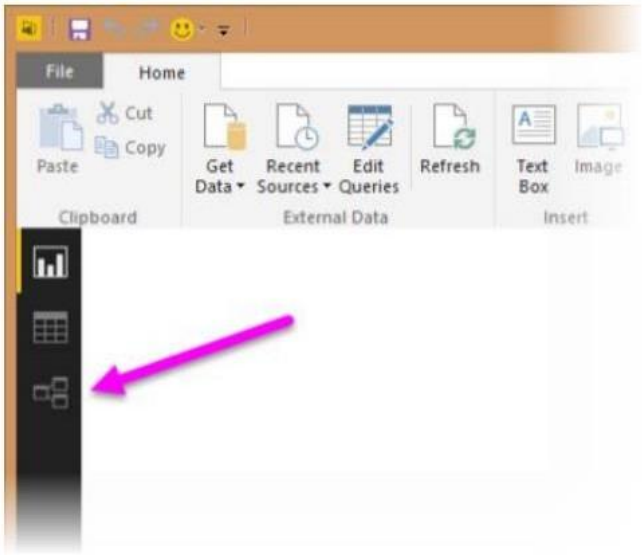
4. Select the New… button



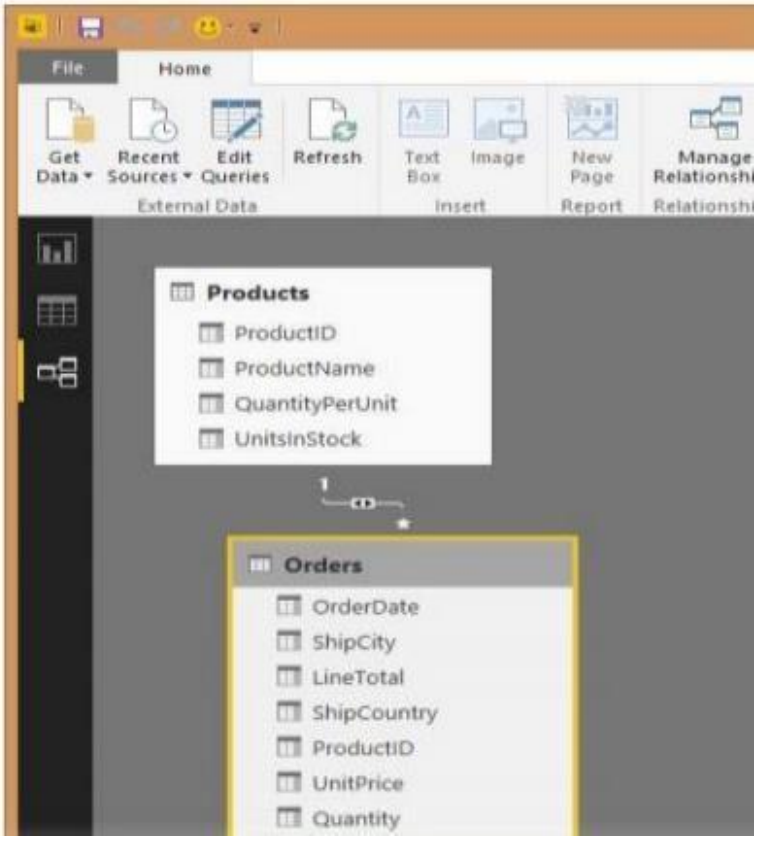
5.When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.



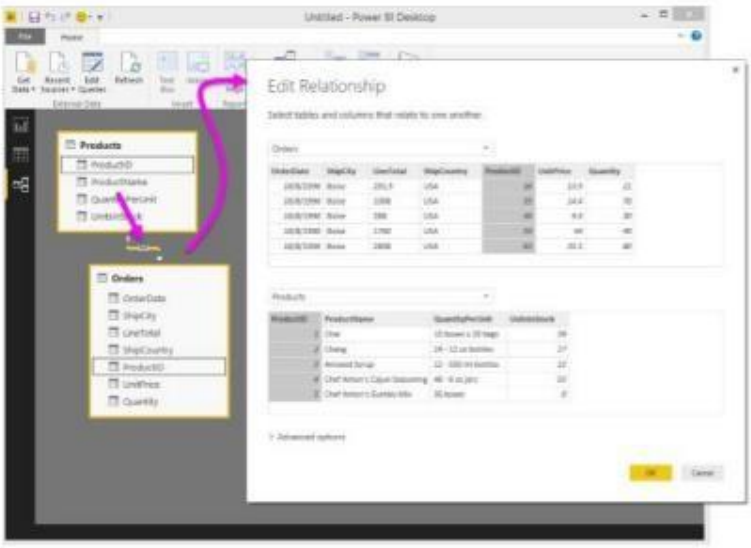
5. Select Cancel, and then select Relationship view in Power BI Desktop.



6. We see the following, which visualizes the relationship between the queries.

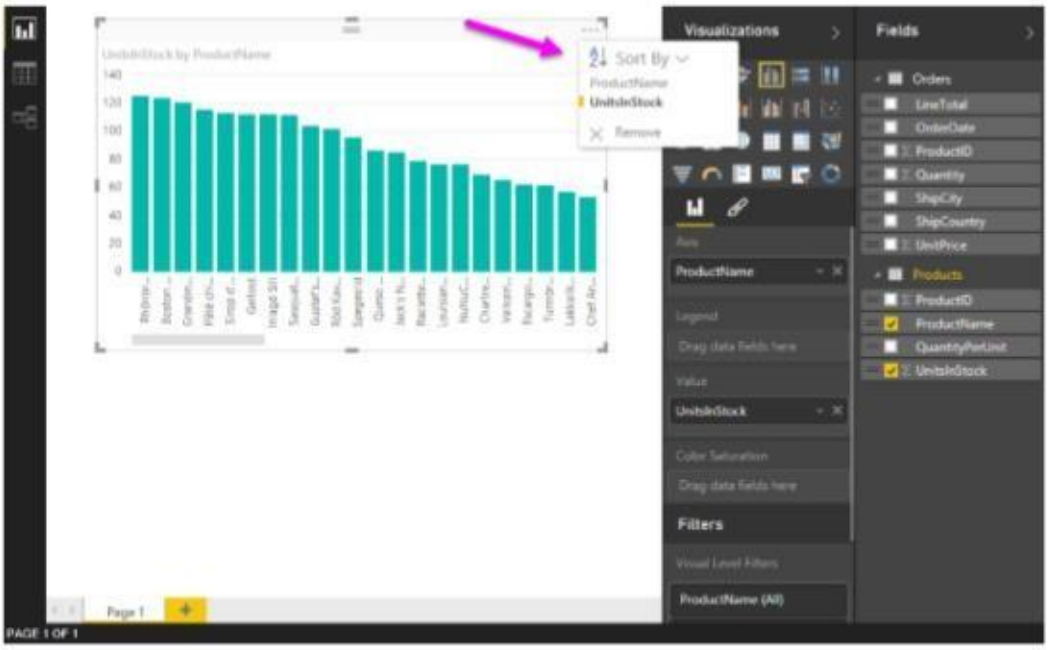


7. When you double-click the arrow on the line that connects the to queries, an Edit Relationship dialog appears.

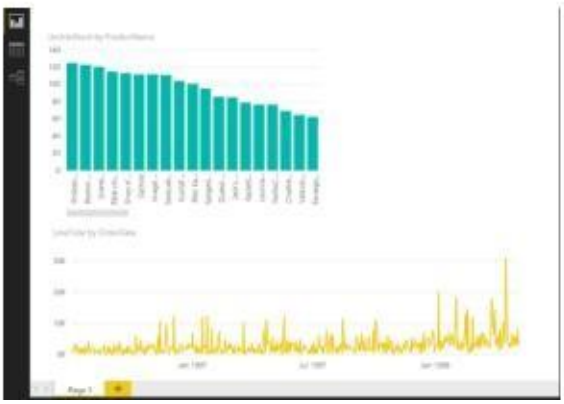


8. No need to make any changes, so we'll just select Cancel to close the Edit Relationship dialog.

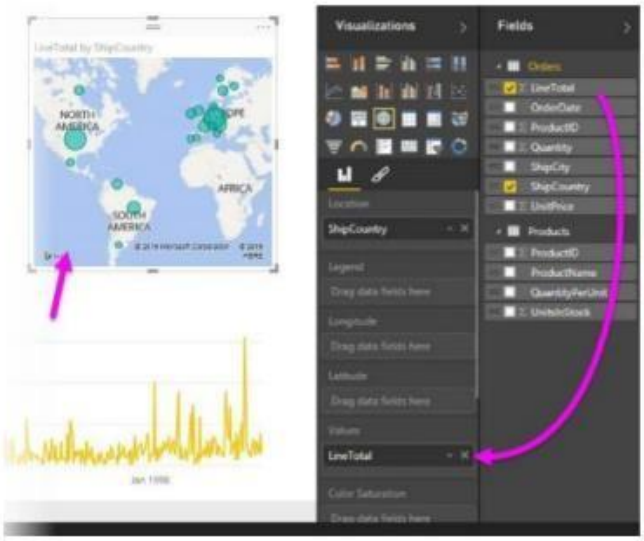
1. Drag UnitsInStock from the Field pane (the Fields pane is along the right of the screen) onto a blank space on the canvas. A Table visualization is created. Next, drag ProductName to the Axis box, found in the bottom half of the Visualizations pane. Then we then select Sort By > UnitsInStock using the skittles in the top right corer of the visualization.



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



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 2: Interact with your report visuals to analyze further

Power BI Desktop lets you interact with visuals that cross-highlight and filter each other to uncover further trends.

1. Click on the light blue circle centered in Canada. Note how the other visuals are filtered to show Stock (ShipCountry) and Total Orders (LineTotal) just for Canada.

PRACTICAL REPORT I

On

Soft Computing

Submitted in fulfilment of the

Requirement for the Award of the Degree of

M.Sc. (Information Technology) - SEM I

By

NAME : AISHWARYA MILIND DESHPANDE
ROLL NO : PSI123002

Department Of Information Technology

THE S.I.A. COLLEGE OF HIGHER EDUCATION

(Affiliated to University of Mumbai)

DOMBIVLI

MAHARASHTRA – 421203

2023– 2024

THE S.I.A COLLEGE OF HIGHER EDUCATION

Affiliated to University of Mumbai

Accredited ‘B+’ Grade

Dombivli(E), Mumbai – 421203

Department of Information Technology

CERTIFICATE

Reaccertified that the experimental work as entered in this journal is as per syllabus in M.Sc. Information Technology for_____as prescribed by University of Mumbai and was done in the Information Technology laboratory of The S.I.A College of Higher Education by the student Mr/Ms_____having Seat No._____of class M.Sc. Information Technology - PART I during the academic year 20 -20.

No. of Experiments completed _____ out of _____

Course Coordinator
Date:

Sign of Incharge **College seal** **Sign of Examiner**

Date: **Date**

INDEX

Sr.No	Practical	Sign
1	**A.** Design a simple linear neural network model. **B.** Calculate the output of neural network net using both binary and bipolar sigmoidal fuction.	
2	**A.** Generate AND/NOT function using McCulloh-Pits neural net. **B.** Generate XOR function using McCulloh-Pits neural net.	
3	**A.** Write a program to implement Hebb's Rule. **B.** Write a program to implement Delta's rule.	
4	**A.** Write a program for Back-Propagation Algorithm. **B.** Write a program for Error Back-Propagation Algorithm.	
5	**A.** Kohonen-Self organizing map. **B.** Adaptive Resonance Theory.	
6	**A.** Write a program for Linear Separation. **B.** Write a program for Hopfield Network Model for Associative Memory.	
7	**A.** Membership and Identity Operators | in, not in. **B.** Membership and Identity Operators | is, is not.	
8	**A.** Find ratios using Fuzzy Logic. **B.** Solve Tipping problem using Fuzzy Logic.	
9	**A.** Write a program for Hopfield Network. **B.** B Write a program for Radial Basis function	
10	**A.** Implementation of Simple genetic algorithm **B.**Create two classes: City and Fitness using Genetic algorithm	

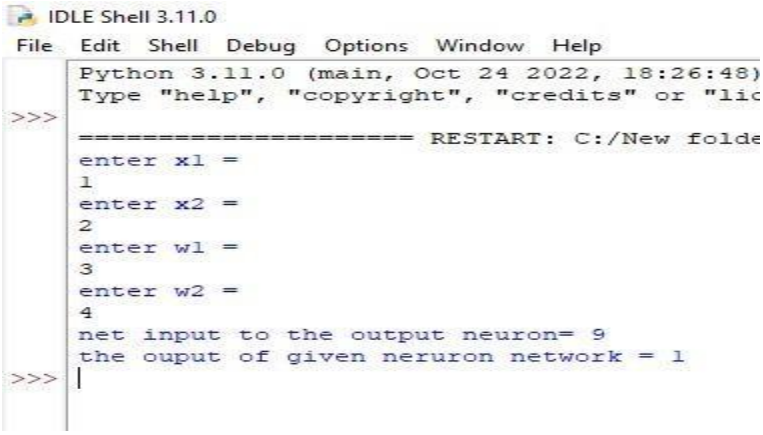
Practical:1

1A: Design and simple linear neural network model.

Code:

```
print("enter x1 =")
x1 =int(input())
print("enter x2 =")
x2 =int(input())
print("enter w1 =")
w1 =int(input())
print("enter w2 =")
w2 =int(input())
yin = x1+x2*w2
print("net input to the output neuron=",yin)
if yin <=0:
yout =0
else:
yout =1
print("the ouput of given neruron network =",yout)
```

Output:

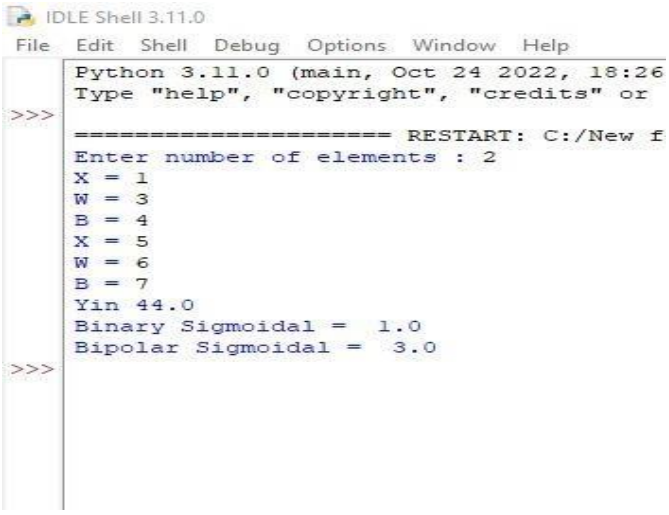


1B: Calculate the output of neural network net using both binary and bipolar sigmoidal function.

Code:

```
 import math
n=int(input("Enter number of elements : "))
yin=0
for i in range(0,n):
    x=float(input("X = "))
    w=float(input("W = "))
    b=float(input("B = "))
    yin = yin + x*w +b
print("Yin" , yin)
binary_sigmoidal = (1 / (1 + (math.e**(-yin))))
print("Binary Sigmoidal = " , round(binary_sigmoidal,3))
bipolar_sigmoidal = (2 / (1 + (math.e**(-yin))))+1
print("Bipolar Sigmoidal = " , round(bipolar_sigmoidal,3))
```

Output:



Practical:2

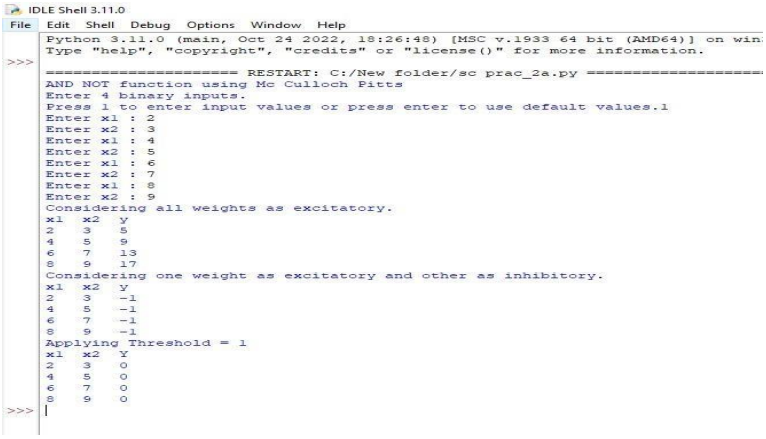
2A: Generate AND/NOT function using McCulloh-pits neural net.

Code:

```
print("AND NOT function using Mc Culloch Pitts")
print("Enter 4 binary inputs.");
x1inputs=[]
x2inputs=[]
c=input("Press 1 to enter input values or press enter to use default values.")
if(c=="1"):
    for i in range(0,4):
        x1=int(input("Enter x1 : "))
        x1inputs.append(x1)
        x2=int(input("Enter x2 : "))
        x2inputs.append(x2)
else:
    x1inputs=[1,1,0,0]
    x2inputs=[1,0,1,0]

print("Considering all weights as excitatory.");
w1 = [1,1,1,1]
w2 = [1,1,1,1]
y=[]
for i in range(0,4):
y.append(x1inputs[i]*w1[i] + x2inputs[i]*w2[i])
print("x1", " x2", " y")
for i in range(0,4):
print(x1inputs[i]," " ,x2inputs[i]," " , y[i])
print("Considering one weight as excitatory and other as inhibitory.");
```

```
w1 = [1,1,1,1]
w2 = [-1,-1,-1,-1]
y=[]
for i in range(0,4):
y.append(x1inputs[i]*w1[i] + x2inputs[i]*w2[i])
print("x1", " x2 ", "y")
for i in range(0,4):     print(x1inputs[i]," ",x2inputs[i]," " , y[i])
print("Applying Threshold = 1")
Y=[]
for i in range(0,4):
   if(y[i]>=1):
      value=1
Y.append(value)
   else:
      value=0
Y.append(value)
print("x1 ", "x2 ", "Y")
for i in range(0,4):
print(x1inputs[i]," ", x2inputs[i]," ",Y[i])
```

Output:



2B: Generate XOR function using McCulloh-pits neural net.

Code:

```
print("XOR function using Mc-Culloch Pitts neuron")
print()
print("Enter 4 binary inputs.");
x1inputs=[]
x2inputs=[]
c=input("Press 1 to enter inputs or Enter to use default inputs.")
if(c=="1"):
    for i in range(0,4):
        x1=int(input("Enter x1 : "))
        x1inputs.append(x1)
        x2=int(input("Enter x2 : "))
        x2inputs.append(x2)
else:
    x1inputs=[1,1,0,0]
    x2inputs=[1,0,1,0]

print("Calculating z1 = x1 x2'")
print("Considering one weight as excitatory and other as inhibitory.");
w1 = [1,1,1,1]
w2 = [-1,-1,-1,-1]
z1=[]
for i in range(0,4):     z1.append(x1inputs[i]*w1[i] + x2inputs[i]*w2[i])
print("x1 " , "x2 " , "z1")
for i in range(0,4):     print(x1inputs[i] ," ", x2inputs[i]," " , z1[i])
print("Calculating z2 = x1' x2")
print("Considering one weight as excitatory and other as inhibitory.");
```

```
w1 = [-1,-1,-1,-1]
w2 = [1,1,1,1]
z2=[]
for i in range(0,4):     z2.append(x1inputs[i]*w1[i] + x2inputs[i]*w2[i])
print("x1 " , "x2 " , "z2")
for i in range(0,4):     print(x1inputs[i] ," " , x2inputs[i] ," ", z2[i])
print("Applying Threshold=1 for z1 and z2")
for i in range(0,4):
    if(z1[i]>=1):        z1[i]=1
    else:        z1[i]=0
    if(z2[i]>=1):        z2[i]=1
    else:        z2[i]=0
print("z1 ","z2")
for i in range(0,4):     print(z1[i] ," ", z2[i])
y = []
v1=1
v2=1
for i in range(0,4):     y.append( z1[i]*v1 + z2[i]*v2 )
print("x1" , "x2" , " y")
for i in range(0,4):     print(x1inputs[i] ," " ,  x2inputs[i] ," ", y[i])
```

Output:

Practical:3 3A:

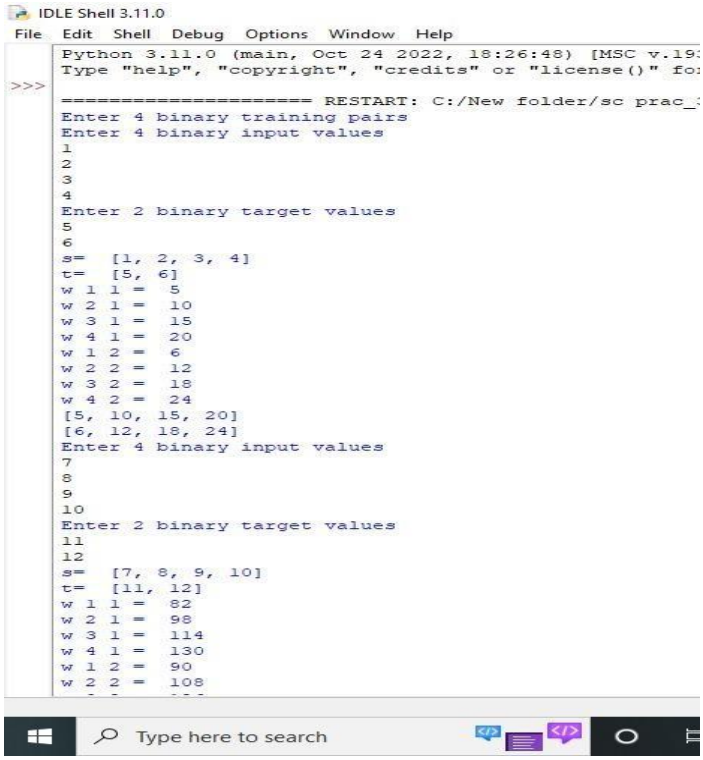
Write a program to implement Hebb's rule.

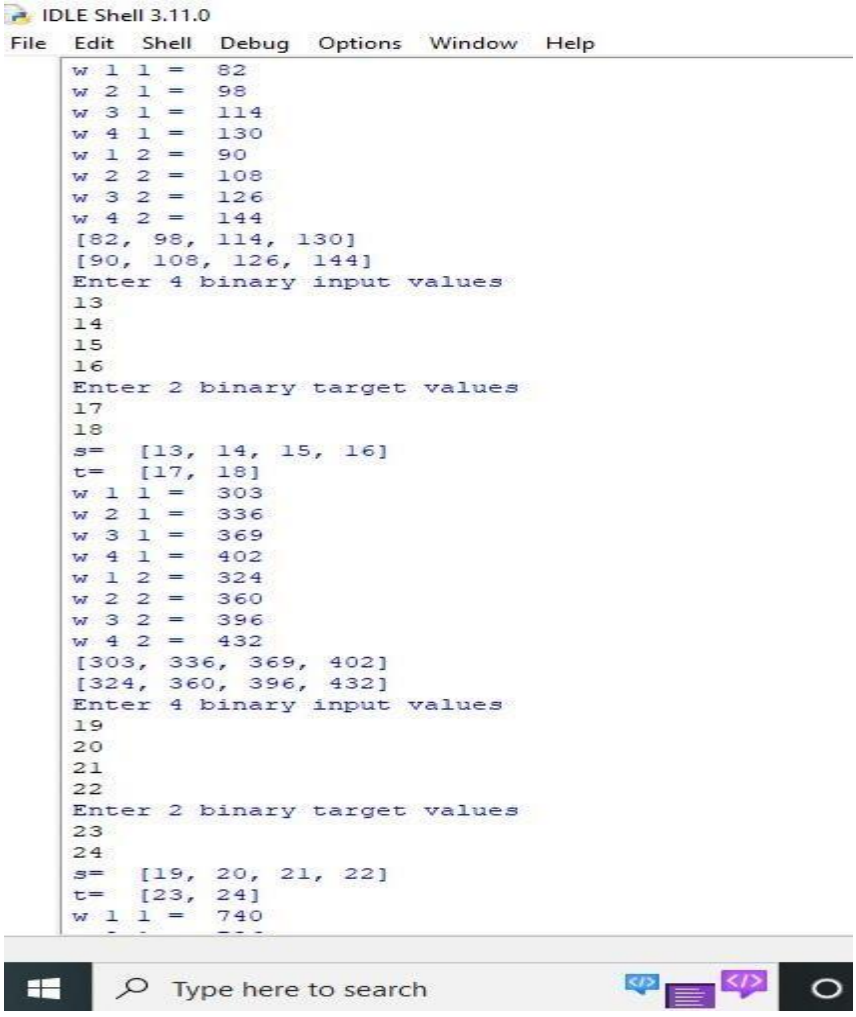
Code:

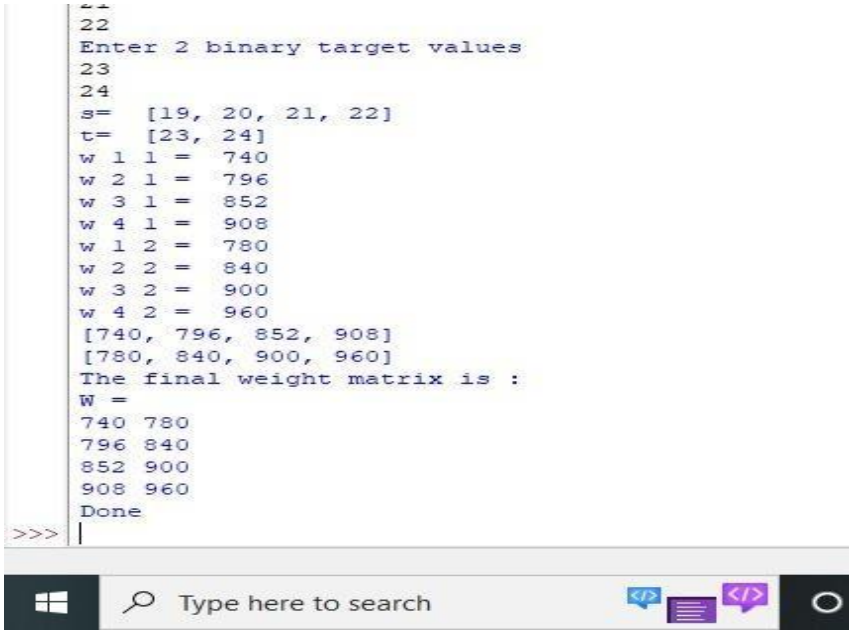
```
print("Enter 4 binary training pairs")
w1=[0,0,0,0]
w2=[0,0,0,0]
for m in range(0,4):
print("Enter 4 binary input values")
    s=[]
    t=[]
    for i in range(0,4):
        x=int(input())
s.append(x)
print("Enter 2 binary target values")
    for i in range(0,2):
       y=int(input())
t.append
print("s= ",s)
print("t= ",t)
    w1new=[]
    for i in range(0,4):
       newweight1=w1[i] + s[i]*t[0]
       w1new.append(newweight1)
    '"print new weights"'
    for i in range(0,4):
        print("w",(i+1),"1 = ",w1new[i])
    w2new=[]
    for i in range(0,4):
```

```
        newweight2=w2[i] + s[i]*t[1]
        w2new.append(newweight2)
    for i in range(0,4):
        print("w",(i+1),"2 = ",w2new[i])
    w1=w1new
    w2=w2new
    print(w1)
    print(w2)
print("The final weight matrix is : ")
print("W = ")
for i in range(0,4):
    print(w1[i] , w2[i])
print("Done")
```

Output:




IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
w 1 1 = 82
w 2 1 = 98
w 3 1 = 114
w 4 1 = 130
w 1 2 = 90
w 2 2 = 108
w 3 2 = 126
w 4 2 = 144
[82, 98, 114, 130]
[90, 108, 126, 144]
Enter 4 binary input values
13
14
15
16
Enter 2 binary target values
17
18
s= [13, 14, 15, 16]
t= [17, 18]
w 1 1 = 303
w 2 1 = 336
w 3 1 = 369
w 4 1 = 402
w 1 2 = 324
w 2 2 = 360
w 3 2 = 396
w 4 2 = 432
[303, 336, 369, 402]
[324, 360, 396, 432]
Enter 4 binary input values
19
20
21
22
Enter 2 binary target values
23
24
s= [19, 20, 21, 22]
t= [23, 24]
w 1 1 = 740
Type here to search


22
Enter 2 binary target values
23
24
s= [19, 20, 21, 22]
t= [23, 24]
w 1 1 = 740
w 2 1 = 796
w 3 1 = 852
w 4 1 = 908
w 1 2 = 780
w 2 2 = 840
w 3 2 = 900
w 4 2 = 960
[740, 796, 852, 908]
[780, 840, 900, 960]
The final weight matrix is :
W =
740 780
796 840
852 900
908 960
Done
>>>
Type here to search

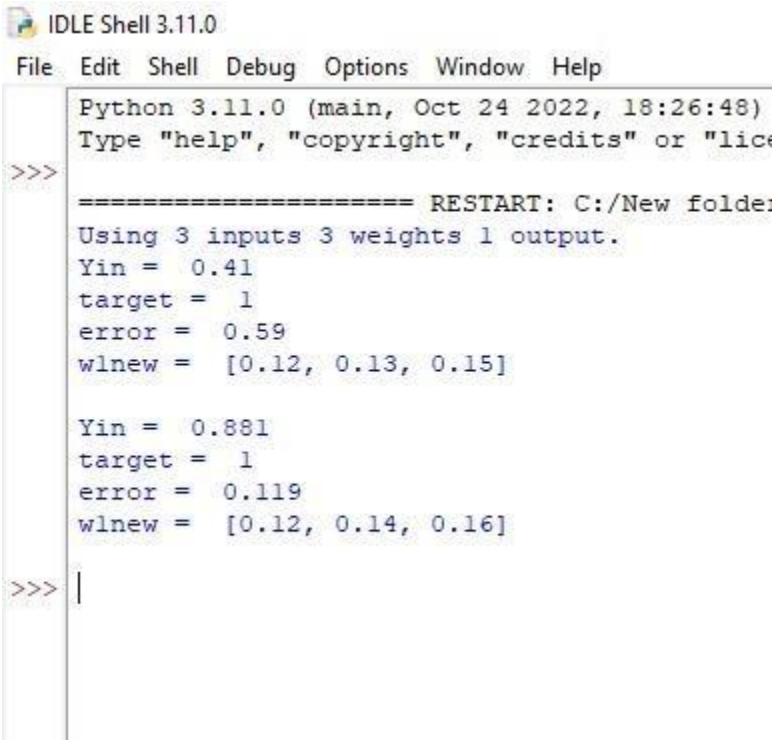
3B: Write a program to implement Delta's rule.

Code:

```
import math
print("Using 3 inputs 3 weights 1 output.")
x1=[0.3,0.5,0.8]  #inputs
w1=[0.1,0.1,0.1] #weights
t=1              #TARGET
a=0.1       #alpha
diff=1      #initial difference
yin=0       #initial net input

while(diff>0.4):
    for i in range(0,3):
        yin = yin + (x1[i]*w1[i])
    yin = yin + 0.25
    yin=round(yin,3)
print("Yin = ",yin)
print("target = ",t)
    diff=t-yin
    diff=round(diff,3)
    diff=math.fabs(diff)
print("error = ",diff)
    neww1=[]
    for i in range(0,3):    #update weights
      w1new=w1[i] + a*diff*x1[i]
      w1new=round(w1new,2)
      neww1.append(w1new)
print("w1new = ",neww1)
```

```
    w1=neww1
print()
```

Output:



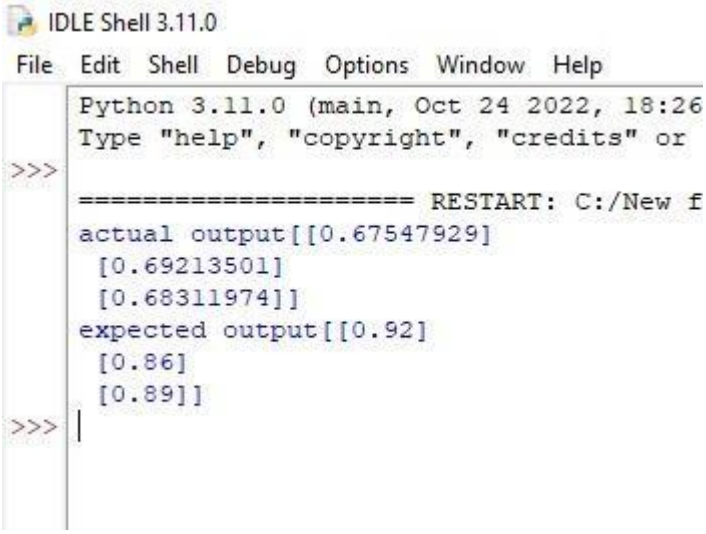
Practical:4

4A: Write a program for Back-Propagation Algorithm.

Code:

```
import numpy as np
X=np.array(([2,9],[1,5],[3,6]),dtype=float)
Y=np.array(([92],[86],[89]),dtype=float)
#scale units
X=X/np.amax(X,axis=0)
Y=Y/100;
class NN(object):
    def _init_(self):
        self.inputsize=2
        self.outputsize=1
        self.hiddensize=3
    self.W1=np.random.randn(self.inputsize,self.hiddensize)
    self.W2=np.random.randn(self.hiddensize,self.outputsize)
    def forward(self,X):
        self.z=np.dot(X,self.W1)
        self.z2=self.sigmoidal(self.z)
        self.z3=np.dot(self.z2,self.W2)
        op=self.sigmoidal(self.z3)
        return op;
    def sigmoidal(self,s):
        return 1/(1+np.exp(-s))
obj=NN() op=obj.forward(X)
print("actual output"+str(op))
print("expected output"+str(Y))
```

Output:


IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26
Type "help", "copyright", "credits" or
>>>
========== RESTART: C:/New f
actual output[[0.67547929]
[0.69213501]
[0.68311974]]
expected output[[0.92]
[0.86]
[0.89]]
>>>

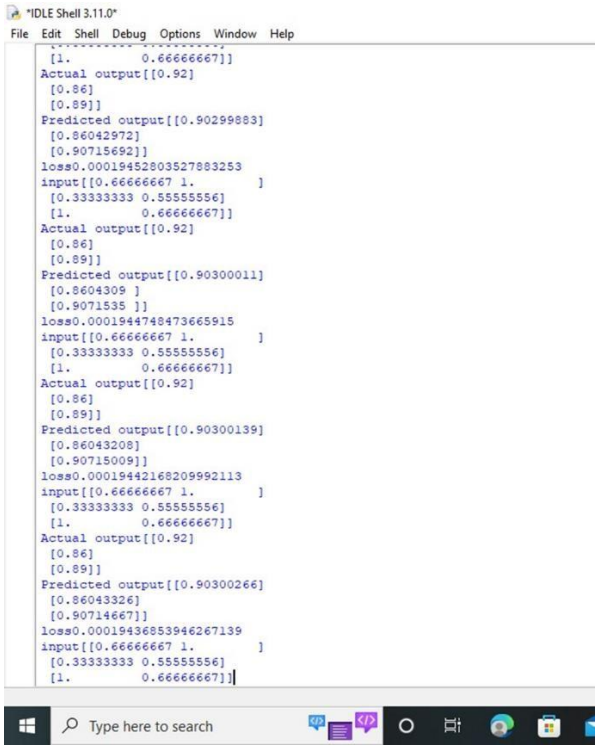
4B: :Write a program for Error Back-Propagation Algorithm.

Code:

```
import numpy as np
X=np.array(([2,9],[1,5],[3,6]),dtype=float)
Y=np.array(([92],[86],[89]),dtype=float)
X=X/np.amax(X,axis=0)
Y=Y/100;
class NN(object):
    def _init_(self):
self.inputsize=2
self.outputsize=1
self.hiddensize=3
self.W1=np.random.randn(self.inputsize,self.hiddensize)
self.W2=np.random.randn(self.hiddensize,self.outputsize)
    def forward(self,X):
self.z=np.dot(X,self.W1)
self.z2=self.sigmoidal(self.z)
self.z3=np.dot(self.z2,self.W2)
        op=self.sigmoidal(self.z3)
        return op;
    def sigmoidal(self,s):
        return 1/(1+np.exp(-s))
    def sigmoidalprime(self,s):
        return s* (1-s)
    def backward(self,X,Y,o):
self.o_error=Y-o
self.o_delta=self.o_error * self.sigmoidalprime(o)
```

```
self.z2_error=self.o_delta.dot(self.W2.T)
self.z2_delta=self.z2_error * self.sigmoidalprime(self.z2)
self.W1 = self.W1 + X.T.dot(self.z2_delta)
self.W2= self.W2+ self.z2.T.dot(self.o_delta)
    def train(self,X,Y):
        o=self.forward(X)
self.backward(X,Y,o)
obj=NN()
for i in range(2000):
    print("input"+str(X))
print("Actual  output"+str(Y))
print("Predicted output"+str(obj.forward(X)))
    print("loss"+str(np.mean(np.square(Y-obj.forward(X)))))
obj.train(X,Y)
```

Output:



Practical:5

5A:Kohonen-self organizing map.

Code:

```
import math
class SOM:
    # Function here computes the winning vector
   # by Euclidean distance
   def winner(self, weights, sample):
       D0 = 0
      D1 = 0
       for i in range(len(sample)):
          D0 = D0 + math.pow((sample[i] - weights[0][i]), 2)
          D1 = D1 + math.pow((sample[i] - weights[1][i]), 2)
          if D0 > D1:
             return 0
          else:
             return 1
    # Function here updates the winning vector
   def update(self, weights, sample, J, alpha):
       for i in range(len(weights)):
          weights[J][i] = weights[J][i] + alpha * (sample[i] - weights[J][i])

      return weights
# Driver code
def main():
    # Training Examples ( m, n )
   T = [[1, 1, 0, 0], [0, 0, 0, 1], [1, 0, 0, 0], [0, 0, 1, 1]]
    m, n = len(T), len(T[0])
    # weight initialization ( n, C )
```

```
    weights = [[0.2, 0.6, 0.5, 0.9], [0.8, 0.4, 0.7, 0.3]]
    # training
ob = SOM()
    epochs = 3
    alpha = 0.5
    for i in range(epochs):
        for j in range(m):
            # training sample
            sample = T[j]
            # Compute winner vector
            J = ob.winner(weights, sample)
            # Update winning vector
            weights = ob.update(weights, sample, J, alpha)
    # classify test sample
    s = [0, 0, 0, 1]
    J = ob.winner(weights, s)
print("Test Sample s belongs to Cluster : ", J)
print("Trained weights : ", weights)
 if _name_ == "_main_":
main()
```

Output:

```
''' ================================ RESTART: D:/New folder/mk/kohonen.py ============================
Test Sample s belongs to Cluster :  0
Trained weights :  [[0.6000000000000001, 0.8, 0.5, 0.9], [0.3333984375, 0.0666015625, 0.7, 0.3]]
>>>
```

5B: Adaptive resonance theory.

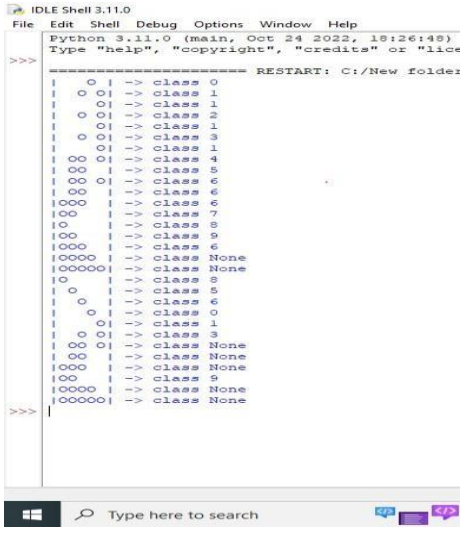
Code:

```
from _future_ import print_function
from _future_ import division
import numpy as np
class ART:
    def _init_(self, n=5, m=10, rho=.5):
        # Comparison layer
self.F1 = np.ones
        # Recognition layer
self.F2 = np.ones(m)
        # Feed-forward weights
self.Wf = np.random.random((m,n))
        # Feed-back weights
self.Wb = np.random.random((n,m))
        # Vigilance
self.rho = rho
        # Number of active units in F2
self.active = 0
    def learn(self, X):
        # Compute F2 output and sort them (I)
self.F2[...] = np.dot(self.Wf, X)
        I = np.argsort(self.F2[:self.active].ravel())[::-1]
        for i in I:
            # Check if nearest memory is above the vigilance level
            d = (self.Wb[:,i]*X).sum()/X.sum()
            if d >= self.rho:
                # Learn data
self.Wb[:,i] *= X
```

```
self.Wf[i,:] = self.Wb[:,i]/(0.5+self.Wb[:,i].sum())
                return self.Wb[:,i], i
        # No match found, increase the number of active units
        # and make the newly active unit to learn data
        if self.active< self.F2.size:
i = self.active
self.Wb[:,i] *= X
self.Wf[i,:] = self.Wb[:,i]/(0.5+self.Wb[:,i].sum())
self.active += 1
            return self.Wb[:,i], i
        return None,None
if _name_ == '_main_':
np.random.seed(1)
    network = ART( 5, 10, rho=0.5)
    data = ["  O ",
"  OO",
            "    O",
"  OO",
            "    O",
"  OO",
            "    O",
            " OO O",
            " OO  ",
            " OO O",
            " OO  ",
            "OOO  ",
            "OO  ",
            "O    ",
            "OO  ",
            "OOO  ",
```

```
            "OOOO ",
            "OOOOO",
            "O    ",
            " O   ",
"  O  ",
            "   O ",
            "    O",
"  OO",
            " OO O",
            " OO  ",
            "OOO  ",
            "OO  ",
            "OOOO ",
            "OOOOO"]
    X = np.zeros(len(data[0]))
    for i in range(len(data)):
        for j in range(len(data[i])):
            X[j] = (data[i][j] == 'O')
        Z, k = network.learn(X)
        print("|%s|"%data[i],"-> class", k)
```

Output:



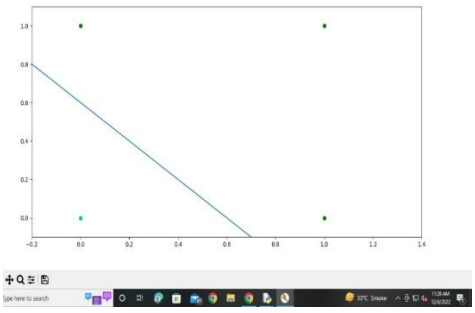
Practical:6

6A: Write a program for Linear Separation.

Code:

```
import matplotlib.pyplot as plt
import numpy as np
fig, ax= plt. subplots()
xmin ,xmax = -0.2,1.4
X = np.arange(xmin,xmax,0.1)
ax.scatter(0,0, color="c")
ax.scatter(0,1, color="g")
ax.scatter(1,0, color="g")
ax.scatter(1,1, color="g")
ax.set_xlim([xmin,xmax])
ax.set_ylim([-0.1,1.1])
m = -1
ax.plot(X, m*X + 0.6, label="decision boundary")
plt.show()
#plt.plot()
```

Output:



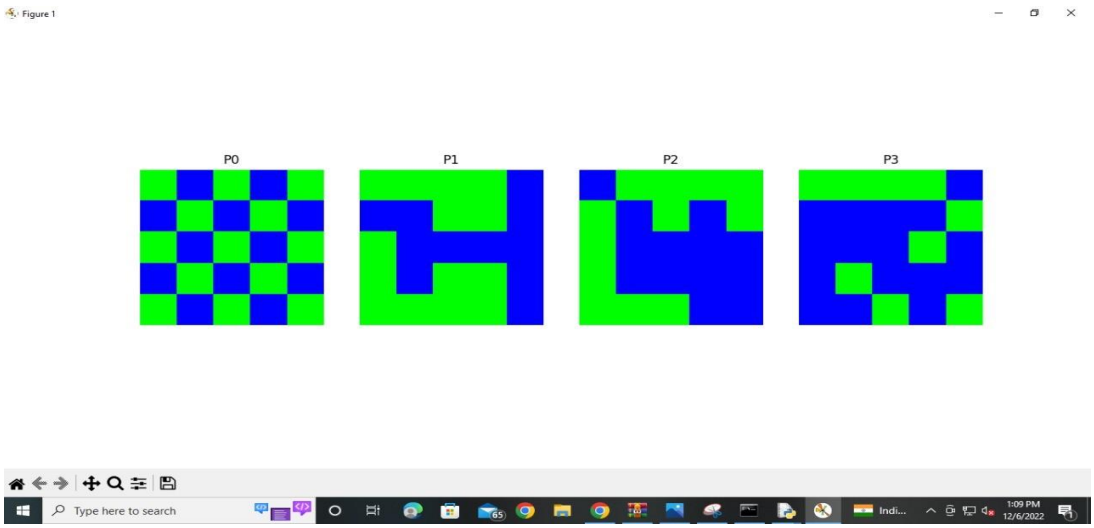
6B: Write a program for Hopfield network model for Associative memory.

Code:

```
import matplotlib.pyplot as plt
from neurodynex.hopfield_network import network, pattern_tools, plot_tools
pattern_size = 5
# create an instance of the class HopfieldNetwork
hopfield_net = network.HopfieldNetwork(nr_neurons= pattern_size**2)
# instantiate a pattern factory
factory = pattern_tools.PatternFactory(pattern_size, pattern_size)
# create a checkerboard pattern and add it to the pattern list
checkerboard = factory.create_checkerboard()
pattern_list = [checkerboard]
# add random patterns to the list
pattern_list.extend(factory.create_random_pattern_list(nr_patterns=3, on_probability=0.5))
plot_tools.plot_pattern_list(pattern_list)
# how similar are the random patterns and the checkerboard? Check the overlaps
overlap_matrix = pattern_tools.compute_overlap_matrix(pattern_list)
plot_tools.plot_overlap_matrix(overlap_matrix)
# let the hopfield network "learn" the patterns. Note: they are not stored
# explicitly but only network weights are updated !
hopfield_net.store_patterns(pattern_list)
# create a noisy version of a pattern and use that to initialize the network
noisy_init_state = pattern_tools.flip_n(checkerboard, nr_of_flips=4)
hopfield_net.set_state_from_pattern(noisy_init_state)
# from this initial state, let the network dynamics evolve.
states = hopfield_net.run_with_monitoring(nr_steps=4)
# each network state is a vector. reshape it to the same shape used to create the patterns.
states_as_patterns = factory.reshape_patterns(states)
# plot the states of the network
```

plot_tools.plot_state_sequence_and_overlap(states_as_patterns, pattern_list, reference_idx=0, suptitle="Network dynamics")

Output:



Practical:7

7A:Membership and identity Operation | in, not in.

Code:

```
list1=[]
c=int(input("Enter the number of elements for list1"))
for i in range(0,c):
ele=int(input("Enter the element:"))
   list1.append(ele)
a=int(input("Enter the number you want to find in list1"))
if a not in list1:
print("The list does not contain ",a)
else:
print("The list contains ",a)
```

Output:

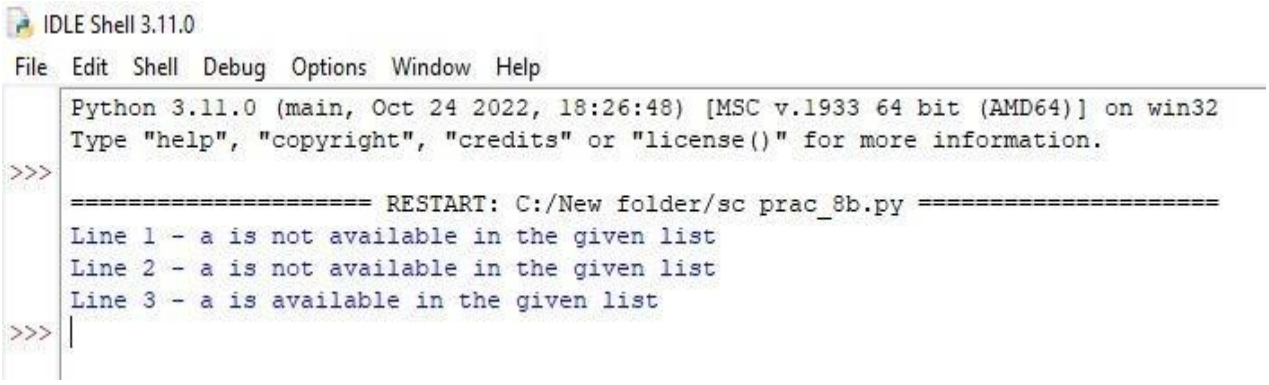
```
=====================================================
Enter the number of elements for list14
Enter the element:3
Enter the element:2
Enter the element:1
Enter the element:5
Enter the number you want to find in list1 0
The list does not contain  0
>>> |
```

7B: Membership and identity Operation | is, is not.

Code:

```
a = 10
b = 20
list = [1,2,3,4,5];
if a in list:
   print ("Line 1 - a is available in the given list")
else:
   print ("Line 1 - a is not available in the given list")
if b not in list:
    print ("Line 2 - a is not available in the given list")
else :
   print ("Line 2 - a is available in the given list")
a = 2
if a in list:
   print ("Line 3 - a is available in the given list")
else:
   print ("Line 3 - a is not available in the given list")
```

Output:



Practical:8

8A:Find Rations using Fuzzy logic.

Code:

```
from fuzzywuzzy import fuzz
from fuzzywuzzy import process
s1="I like SC"
s2="I like HC"
print ("FuzzyWuzzy Ratio:", fuzz.ratio(s1, s2))
print ("FuzzyWuzzyPartialRatio:", fuzz.partial_ratio(s1, s2))
print ("FuzzyWuzzyTokenSortRatio:", fuzz.token_sort_ratio(s1, s2))
print ("FuzzyWuzzyTokenSetRatio:", fuzz.token_set_ratio(s1, s2))
print ("FuzzyWuzzyWRatio:", fuzz.Wratio(s1, s2),'\n\n')
```

Output:

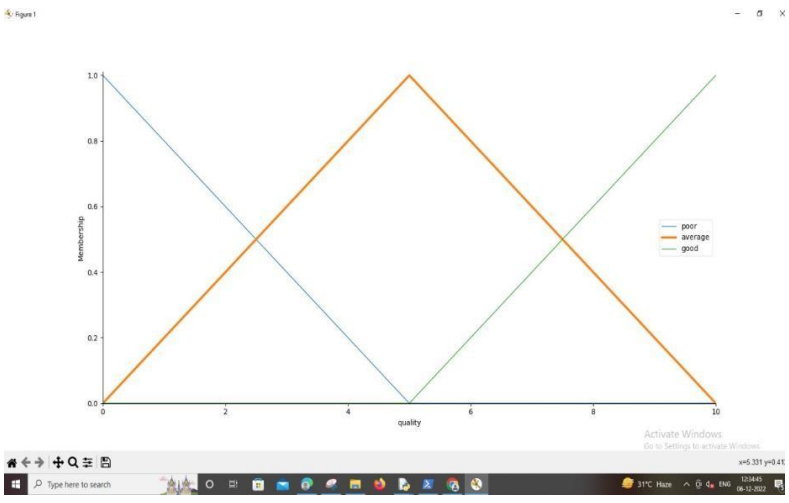
```
FuzzyWuzzy Ratio: 89
FuzzyWuzzy PartialRatio: 89
FuzzyWuzzy TokenSortRatio: 67
FuzzyWuzzy TokenSetRatio: 89
```

8B: Solve Tipping problem using Fuzzy logic.

Code:

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
# New Antecedent/Consequent objects hold universe variables and membership
# functions
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')
# Auto-membership function population is possible with .automf(3, 5, or 7)
quality.automf(3)
service.automf(3)
# Custom membership functions can be built interactively with a familiar,
# Pythonic API
tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25]
```

Output:



```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:3
Type "help", "copyright", "credits" or "license()" for mo
>>>
========== RESTART: D:/New folder/mk/newpract9.
>>> 9
9
>>> 11
11
>>>
========== RESTART: D:/New folder/mk/tipping.p
>>> quality['average'].view()
>>>
```

Practical 9

9A) Write a program for Hopfield Network.

Algorithm

Step 0. Initialize activations of all units.
Initialize Δt to a small value.

Step 1. While the stopping condition is false, do Steps 2–6.

Step 2. Perform Steps 3–5 n^2 times (n is the number of cities).

Step 3. Choose a unit at random.

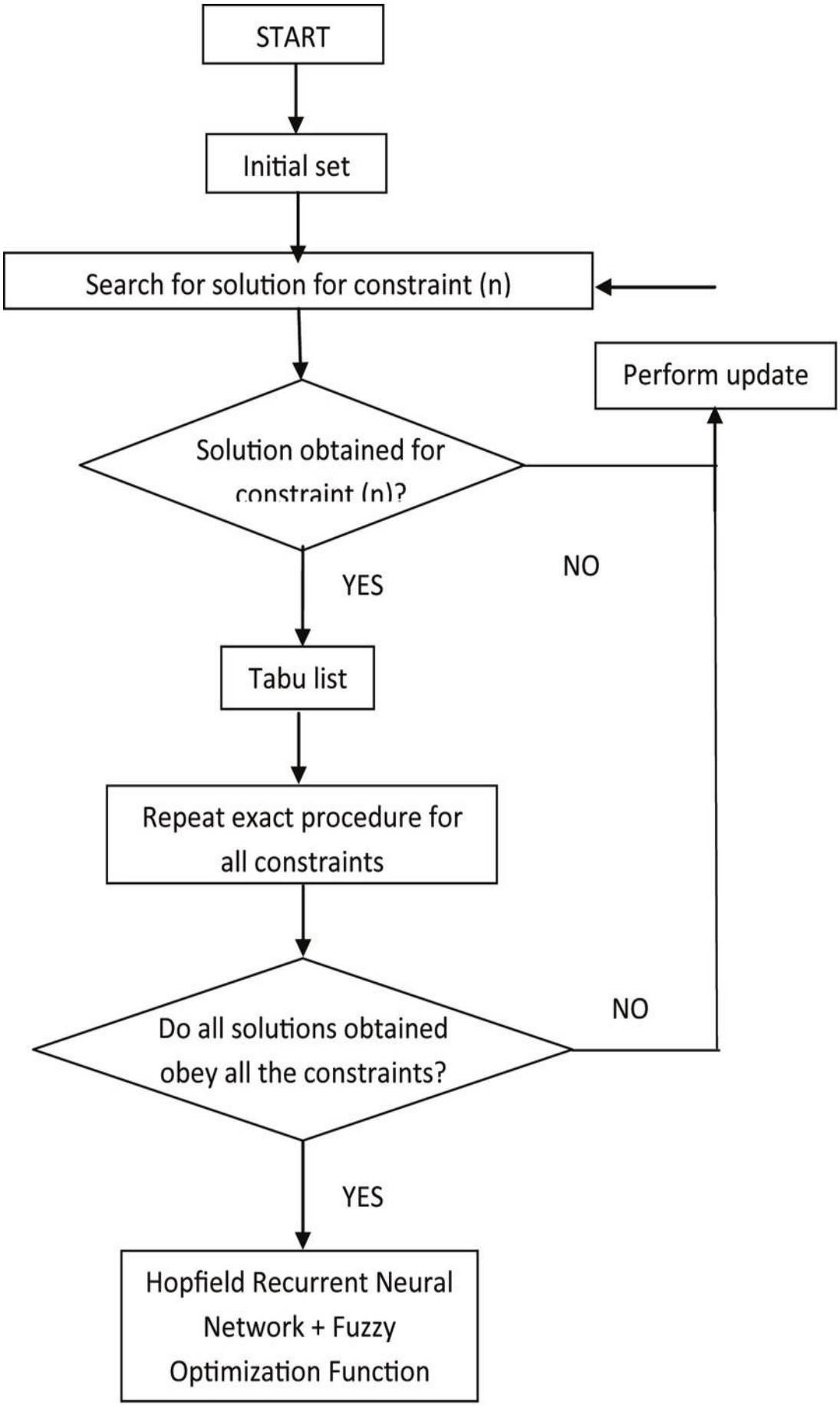
Step 4. Change activity on selected unit:

$$u_{x,i}(\text{new}) = u_{x,i}(\text{old}) + \Delta t[-u_{x,i}(\text{old}) - A\sum_{j\neq i} v_{x,j} - B\sum_{y\neq x} v_{y,i} + C\{N - \sum_{x}\sum_{j} v_{x,j}\} - D\sum_{y\neq x} d_{x,y}(v_{y,i+1} + v_{y,i-1})].$$

Step 5. Apply output function:

$$v_{x,i} = 0.5[1 + \tanh(\alpha u_{x,i})].$$

Step 6. Check stopping condition.


START
Initial set
Search for solution for constraint (n)
Perform update
Solution obtained for constraint (n)?
NO
YES
Tabu list
Repeat exact procedure for all constraints
Do all solutions obtained obey all the constraints?
NO
YES
Hopfield Recurrent Neural Network + Fuzzy Optimization Function

```
#include "hop.h"
neuron::neuron(int *j)
{
int i;
for(i=0;i<4;i++)
    {
    weightv[i]= *(j+i);
    }
}
int neuron::act(int m, int *x)
{
int  i;
int a=0;
for(i=0;i<m;i++)
    {
    a += x[i]*weightv[i];
    }
return a;
}
int network::threshld(int k)
{
if(k>=0)
    return (1);
else
    return (0);
}
network::network(int a[4],int b[4],int c[4],int d[4])
{
nrn[0] = neuron(a) ;
nrn[1] = neuron(b) ;
nrn[2] = neuron(c) ;
nrn[3] = neuron(d) ;
}



void network::activation(int *patrn)
{
int i,j;
for(i=0;i<4;i++)
    {
    for(j=0;j<4;j++)
        {
        cout<<"\n nrn["<<i<<"].weightv["<<j<<"] is "
            <<nrn[i].weightv[j];
        }
    nrn[i].activation = nrn[i].act(4,patrn);
    cout<<"\nactivation is "<<nrn[i].activation;
    output[i]=threshld(nrn[i].activation);
```

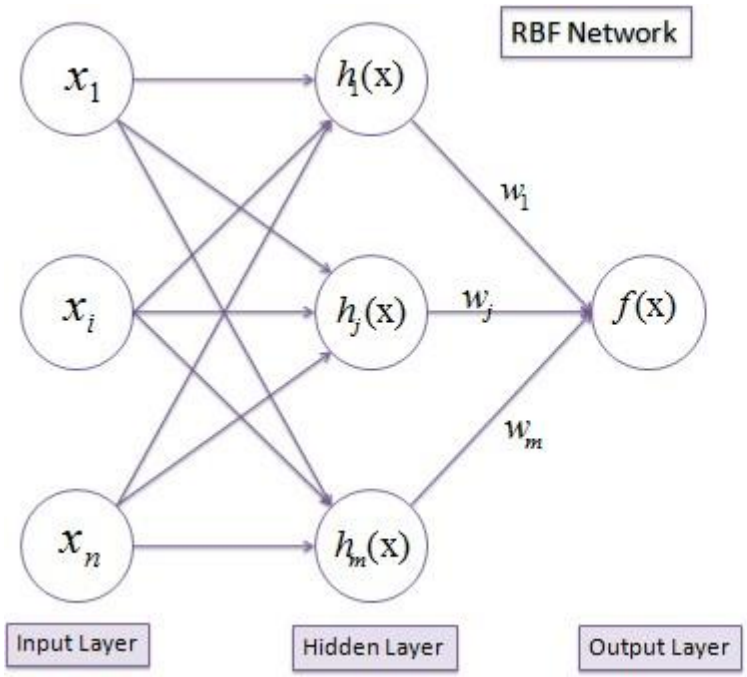
```
    cout<<"\noutput value is  "<<output[i]<<"\n";
    }
}
void main ()
{
int patrn1[]= {1,0,1,0},i;
int wt1[]= {0,-3,3,-3};
int wt2[]= {-3,0,-3,3};
int wt3[]= {3,-3,0,-3};
int wt4[]= {-3,3,-3,0};
cout<<"\nTHIS PROGRAM IS FOR A HOPFIELD NETWORK WITH A SINGLE LAYER
OF";
cout<<"\n4 FULLY INTERCONNECTED NEURONS. THE NETWORK SHOULD
RECALL
THE";
cout<<"\nPATTERNS 1010 AND 0101 CORRECTLY.\n";
//create the network by calling its constructor.
// the constructor calls neuron constructor as many times as the number of
// neurons in the network.
network h1(wt1,wt2,wt3,wt4);
//present a pattern to the network and get the activations of the neurons
h1.activation(patrn1);
//check if the pattern given is correctly recalled and give message
for(i=0;i<4;i++)
    {
    if (h1.output[i] == patrn1[i])
        cout<<"\n pattern= "<<patrn1[i]<<
        " output = "<<h1.output[i]<<" component matches";
    else
        cout<<"\n pattern= "<<patrn1[i]<<
        "  output = "<<h1.output[i]<<
        "  discrepancy occurred";
    }
cout<<"\n\n";
int patrn2[]= {0,1,0,1};
h1.activation(patrn2);
for(i=0;i<4;i++)
    {
    if (h1.output[i] == patrn2[i])
        cout<<"\n pattern= "<<patrn2[i]<<
        " output = "<<h1.output[i]<<" component matches";
    else
        cout<<"\n pattern= "<<patrn2[i]<<
        "  output = "<<h1.output[i]<<
        "  discrepancy occurred";
     }
}
======== End code of main program==========
//Hop.h
//Single layer Hopfield Network with 4 neurons
```

```
#include <stdio.h>
#include <iostream.h>
#include <math.h>
class neuron
{
protected:
    int activation;
    friend class network;
public:
    int weightv[4];
    neuron() {};
    neuron(int *j) ;
    int act(int, int*);
};
class network
{
public:
    neuron nrn[4];
    int output[4];
    int threshld(int) ;
    void activation(int j[4]);
    network(int*,int*,int*,int*);
};
```

Practical 9b: Write a program for Radial Basis function

Radial Basis Function Networks (RBF)

RBF networks have three layers: input layer, hidden layer and output layer. One neuron in the input layer corresponds to each predictor variable. With respects to categorical variables, *n*-1 neurons are used where *n* is the number of categories. Hidden layer has a variable number of neurons. Each neuron consists of a radial basis function centered on a point with the same dimensions as the predictor variables. The output layer has a weighted sum of outputs from the hidden layer to form the network outputs.



$$f(\mathrm{x}) = \sum_{j=1}^{m} w_j h_j(\mathrm{x})$$

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right)$$

Algorithm

$h(x)$ is the Gaussian activation function with the parameters r (the radius or standard deviation) and c (the center or average taken from the input space) defined separately at each RBF unit. The learning process is based on adjusting the parameters of the network to reproduce a set of input-output patterns. There are three types of parameters; the weight w between the hidden nodes and the output nodes, the center c of each neuron of the hidden layer and the unit width r.

Unit Center (*c*)

Any clustering algorithm can be used to determine the RBF unit centers (e.g., K-means clustering). A set of clusters each with r-dimensional centers is determined by the number of input variables or nodes of the input layer. The cluster centers become the centers of the RBF units. The number of clusters, H, is a design parameter and determines the number of nodes in the hidden layer. The K-means clustering algorithm proceeds as follows:

1. Initialize the center of each cluster to a different randomly selected training pattern.
2. Assign each training pattern to the nearest cluster. This can be accomplished by calculating the Euclidean distances between the training patterns and the cluster centers.
3. When all training patterns are assigned, calculate the average position for each cluster center. They then become new cluster centers.
4. Repeat steps 2 and 3, until the cluster centers do not change during the subsequent iterations.

Unit width (*r*)

When the RBF centers have been established, the width of each RBF unit can be calculated using the K-nearest neighbors algorithm. A number K is chosen, and for each center, the K nearest centers is found. The root-mean squared distance between the current cluster center and its K nearest neighbors is calculated, and this is the value chosen for the unit width (r). So, if the current cluster center is c_j, the r value is:

$$r_j = \sqrt{\frac{\sum_{i=1}^{k} (c_j - c_i)^2}{k}}$$

A typical value for K is 2, in which case s is set to be the average distance from the two nearest neighboring cluster centers.

Weights (*w*)

Using the linear mapping, **w** vector is calculated using the output vector (**y**) and the design matrix **H**.

$$y = wH$$
$$w = (H'H)\, H'y$$

The basis functions are (unnormalized) gaussians, the output layer is linear and the weights are learned by a simple pseudo-inverse.

```
from scipy import *
from scipy.linalg import norm, pinv

from matplotlib import pyplot as plt

class RBF:

def__init_(self, indim, numCenters, outdim):
        self.indim = indim
        self.outdim = outdim
        self.numCenters = numCenters
        self.centers = [random.uniform(-1, 1, indim) for i in xrange(numCenters)]
        self.beta = 8
        self.W = random.random((self.numCenters, self.outdim))

    def _basisfunc(self, c, d):
```

```
        assert len(d) == self.indim
        return exp(-self.beta * norm(c-d)**2)

    def _calcAct(self, X):
        # calculate activations of RBFs
        G = zeros((X.shape[0], self.numCenters), float)
        for ci, c in enumerate(self.centers):
            for xi, x in enumerate(X):
                G[xi,ci] = self._basisfunc(c, x)
        return G

    def train(self, X, Y):
        """ X: matrix of dimensions n x indim
            y: column vector of dimension n x 1 """

        # choose random center vectors from training set
        rnd_idx = random.permutation(X.shape[0])[:self.numCenters]
        self.centers = [X[i,:] for i in rnd_idx]

        print "center", self.centers
        # calculate activations of RBFs
        G = self._calcAct(X)
        print G

        # calculate output weights (pseudoinverse)
        self.W = dot(pinv(G), Y)

    def test(self, X):
        """ X: matrix of dimensions n x indim """

        G = self._calcAct(X)
        Y = dot(G, self.W)
        return Y


if__name__== '__main__':
    #-------1D Example ----------------------------------------------------------
    n = 100

    x = mgrid[-1:1:complex(0,n)].reshape(n, 1)
    # set y and add random noise
    y = sin(3*(x+0.5)**3 - 1)
    # y += random.normal(0, 0.1, y.shape)

    # rbf regression
    rbf = RBF(1, 10, 1)
    rbf.train(x, y)
    z = rbf.test(x)

    # plot original data
```

```
plt.figure(figsize=(12, 8))
plt.plot(x, y, 'k-')

# plot learned model
plt.plot(x, z, 'r-', linewidth=2)

# plot rbfs
plt.plot(rbf.centers, zeros(rbf.numCenters), 'gs')

for c in rbf.centers:
    # RF prediction lines
    cx = arange(c-0.7, c+0.7, 0.01)
    cy = [rbf._basisfunc(array([cx_]), array([c])) for cx_ in cx]
    plt.plot(cx, cy, '-', color='gray', linewidth=0.2)

plt.xlim(-1.2, 1.2)
plt.show()
```

Using R

Radial Basis Functions

A radial basis function, RBF, $\phi(x)\phi(x)$ is a function with respect to the origin or a certain point cc, ie, $\phi(x)=f(\|x-c\|)\phi(x)=f(\|x-c\|)$ where the norm is usually the Euclidean norm but can be other type of measure.
The RBF learning model assumes that the dataset $D=(xn,yn),n=1\dots N$ $D=(xn,yn),n=1\dots N$ influences the hypothesis set $h(x)h(x)$, for a new observation xx, in the following way:

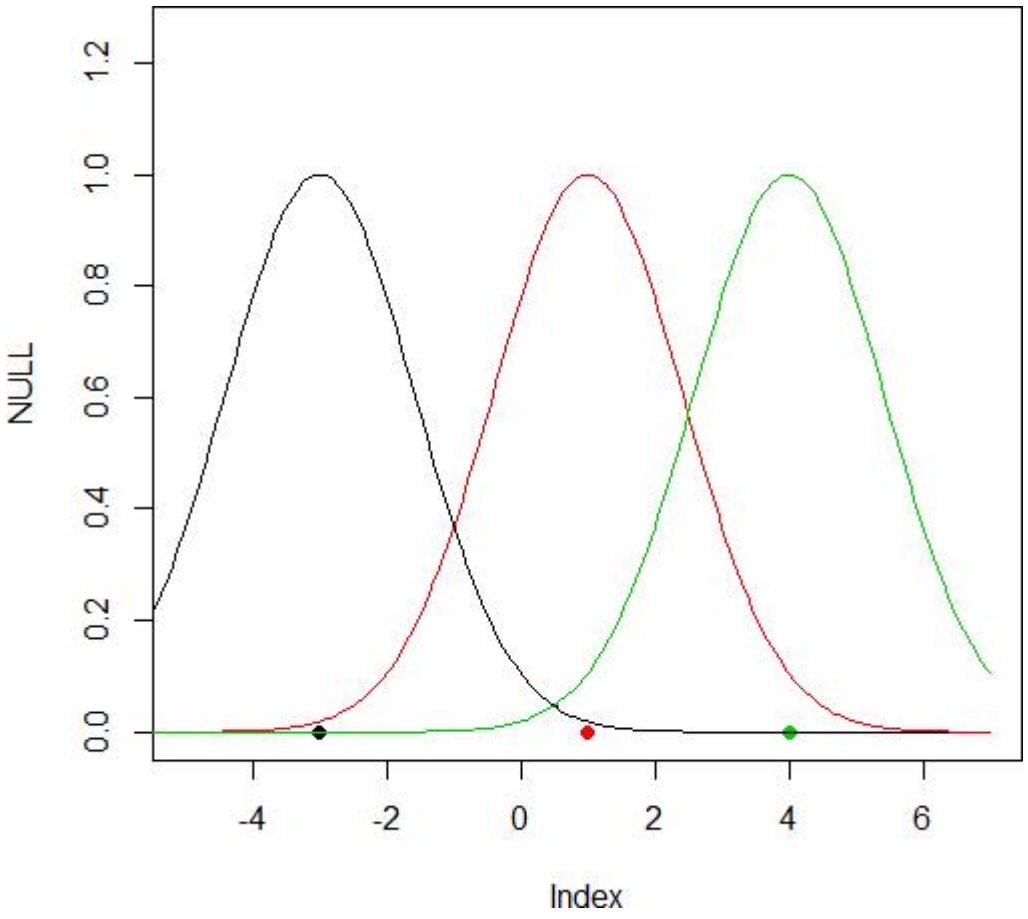
$$h(x)=\sum n=1Nwn\times\exp(-\gamma\|x-xn\|2)h(x)=\sum n=1Nwn\times\exp(-\gamma\|x-xn\|2)$$

which means that each xixi of the dataset influences the observation in a gaussian shape. Of course, if a datapoint is far away from the observation its influence is residual (the exponential decay of the tails of the gaussian make it so). It is an example of a localized function $(x\to\infty\Longrightarrow\phi(x)\to0x\to\infty\Longrightarrow\phi(x)\to0)$. Notice that other type of radial functions can be used. Some egs:

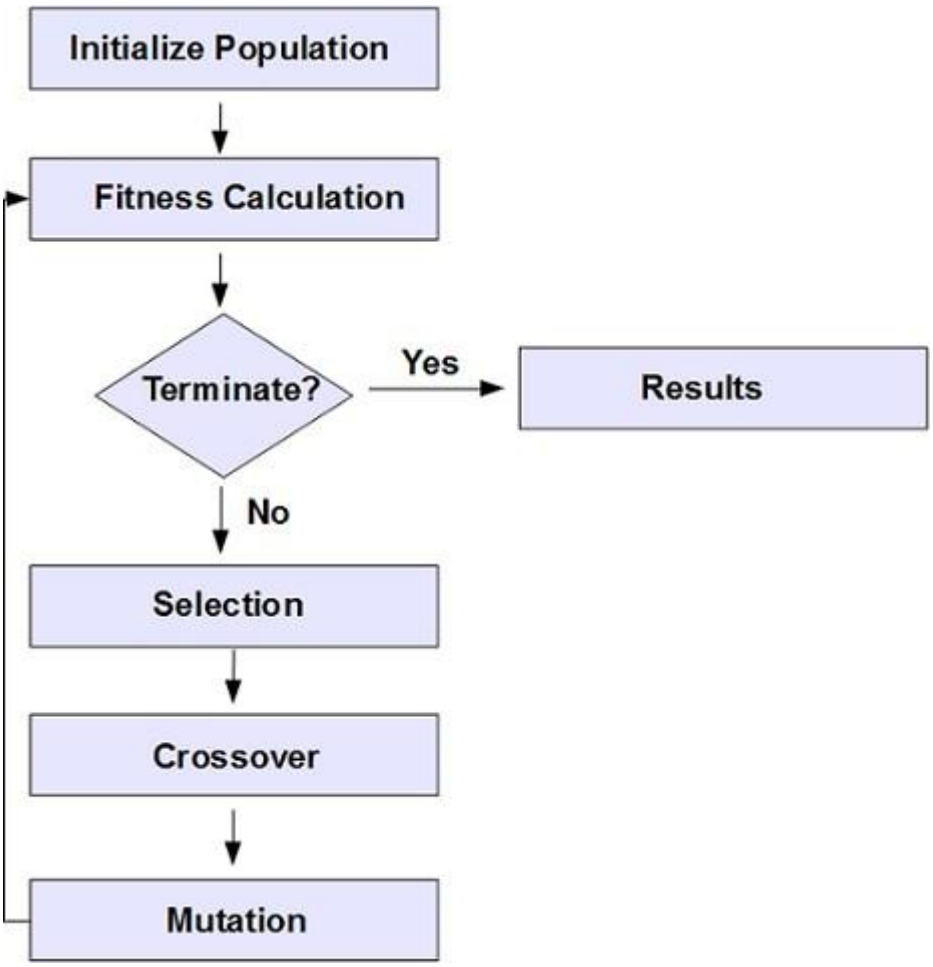
- Multi-quadratic: $\phi(x)=x2+\gamma2-------\sqrt{}\phi(x)=x2+\gamma2$
- Thin plate spline: $\phi(x)=x2\ln(x)$

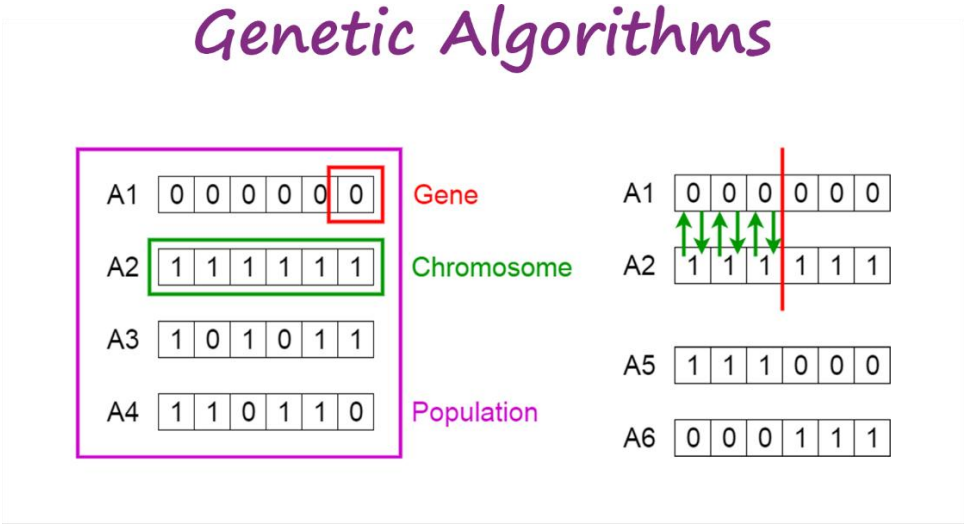
```
rbf.gauss <- function(gamma=1.0) {

  function(x) {
    exp(-gamma * norm(as.matrix(x),"F")^2 )
  }
}
```

```r
D <- matrix(c(-3,1,4), ncol=1) # 3 datapoints
N <- length(D[,1])



xlim <- c(-5,7)
plot(NULL,xlim=xlim,ylim=c(0,1.25),type="n")
points(D,rep(0,length(D)),col=1:N,pch=19)
x.coord = seq(-7,7,length=250)
gamma <- 1.5
for (i in 1:N) {
  points(x.coord, lapply(x.coord - D[i,], rbf.gauss(gamma)), type="l", col=i)
}
plot(NULL,xlim=xlim,ylim=c(0,1.25),type="n")
points(D,rep(0,length(D)),col=1:N,pch=19)
x.coord = seq(-7,7,length=250)
gamma <- 0.25
for (i in 1:N) {
  points(x.coord, lapply(x.coord - D[i,], rbf.gauss(gamma)), type="l", col=i)
```


NULL
0.0
0.2
0.4
0.6
0.8
1.0
1.2
-4
-2
0
2
4
6
Index

10A) Implementation of Simple genetic algorithm





```
import random

# Number of individuals in each generation
POPULATION_SIZE = 100

# Valid genes
GENES = '''abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP
QRSTUVWXYZ 1234567890, .-;:_!"#%&/()=?@${[]}'''

# Target string to be generated
TARGET = "I love GeeksforGeeks"

class Individual(object):
    '''
    Class representing individual in population
    '''
    def __init_(self, chromosome):
        self.chromosome = chromosome
        self.fitness = self.cal_fitness()

    @classmethod
    def mutated_genes(self):
        '''
        create random genes for mutation
        '''
        global GENES
        gene = random.choice(GENES)
        return gene

    @classmethod
    def create_gnome(self):
        '''
        create chromosome or string of genes
        '''
        global  TARGET
        gnome_len = len(TARGET)
        return [self.mutated_genes() for _ in range(gnome_len)]

    def mate(self, par2):
        '''
        Perform mating and produce new offspring
        '''

        # chromosome for offspring
        child_chromosome = []
        for gp1, gp2 in zip(self.chromosome, par2.chromosome):

            # random probability
            prob = random.random()
```

```
            # if prob is less than 0.45, insert gene
            # from parent 1
            if prob < 0.45:
                child_chromosome.append(gp1)

            # if prob is between 0.45 and 0.90, insert
            # gene from parent 2
            elif prob < 0.90:
                child_chromosome.append(gp2)

            # otherwise insert random gene(mutate),
            # for maintaining diversity
            else:
                child_chromosome.append(self.mutated_genes())

        # create new Individual(offspring) using
        # generated chromosome for offspring
        return Individual(child_chromosome)

    def cal_fitness(self):
        '''
        Calculate fittness score, it is the number of
        characters in string which differ from target
        string.
        '''
        global TARGET
        fitness = 0
        for gs, gt in zip(self.chromosome, TARGET):
            if gs != gt: fitness+= 1
        return fitness

# Driver code
def main():
    global POPULATION_SIZE

    #current generation
    generation = 1

    found = False
    population = []

    # create initial population
    for _ in range(POPULATION_SIZE):
                gnome = Individual.create_gnome()
                population.append(Individual(gnome))

    while not found:

        # sort the population in increasing order of fitness score
        population = sorted(population, key = lambda x:x.fitness)
```

```
        # if the individual having lowest fitness score ie.
        # 0 then we know that we have reached to the target
        # and break the loop
        if population[0].fitness <= 0:
            found = True
            break

        # Otherwise generate new offsprings for new generation
        new_generation = []

        # Perform Elitism, that mean 10% of fittest population
        # goes to the next generation
        s = int((10*POPULATION_SIZE)/100)
        new_generation.extend(population[:s])

        # From 50% of fittest population, Individuals
        # will mate to produce offspring
        s = int((90*POPULATION_SIZE)/100)
        for _ in range(s):
            parent1 = random.choice(population[:50])
            parent2 = random.choice(population[:50])
            child = parent1.mate(parent2)
            new_generation.append(child)

        population = new_generation

        print("Generation: {}\tString: {}\tFitness: {}".\
              format(generation,
              "".join(population[0].chromosome),
              population[0].fitness))

        generation += 1


    print("Generation: {}\tString: {}\tFitness: {}".\
          format(generation,
          "".join(population[0].chromosome),
          population[0].fitness))

if__name__== '_main_':
    main()
```

Output:

```
Generation: 1    String: tO{"-?=jH[k8=B4]Oe@}    Fitness: 18

Generation: 2    String: tO{"-?=jH[k8=B4]Oe@}    Fitness: 18

Generation: 3    String: .#lRWf9k_Ifslw #O$k_    Fitness: 17
```

```
Generation: 4    String: .-1Rq?9mHqk3Wo]3rek_    Fitness: 16
Generation: 5    String: .-1Rq?9mHqk3Wo]3rek_    Fitness: 16
Generation: 6    String: A#ldW) #lIkslw cVek)    Fitness: 14
Generation: 7    String: A#ldW) #lIkslw cVek)    Fitness: 14
Generation: 8    String: (, o x _x%Rs=, 6Peek3    Fitness: 13
                 .
                 .
                 .
Generation: 29    String: I lope Geeks#o, Geeks    Fitness: 3
Generation: 30    String: I loMe GeeksfoBGeeks    Fitness: 2
Generation: 31    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 32    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 33    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 34    String: I love GeeksforGeeks    Fitness: 0
```

10B)Create two classes: City and Fitness using Genetic algorithm

first create a City class that will allow us to create and handle our cities.

```
class
City:
        def__init__(self, x, y):
            self.x = x
            self.y = y

        def distance(self, city):
            xDis = abs(self.x - city.x)
            yDis = abs(self.y - city.y)
            distance = np.sqrt((xDis ** 2) + (yDis ** 2))
            return distance

        def__repr__(self):
            return "(" + str(self.x) + "," + str(self.y) + ")"
```

```
class
Fitness:
            def__init__(self, route):
                self.route = route
                self.distance = 0
                self.fitness= 0.0

            def routeDistance(self):
                if self.distance ==0:
                    pathDistance = 0
                    for i in range(0, len(self.route)):
                        fromCity = self.route[i]
                        toCity = None
                        if i + 1 < len(self.route):
                            toCity = self.route[i + 1]
                        else:
                            toCity = self.route[0]
                        pathDistance += fromCity.distance(toCity)
                    self.distance = pathDistance
                return self.distance

            def routeFitness(self):
```

```
        if self.fitness == 0:
            self.fitness = 1 / float(self.routeDistance())
        return self.fitness
```

Create Population

https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35