INDEX

Sr. No.	Title	Page No.	Sign
Pr 1	Write the programs for the following: 1 a) Text Delimited CSV to HORUS format 1 b) XML to HORUS format 1 c) JSON to HORUS format 1 d) MySql database to HORUS format 1 e) Picture(JPEG) to HORUS format 1 f) Video to HORUS format 1 g) Audio to HORUS format	3	
Pr 2	Write the programs for the following: 2 a) Fixers Utilities 2 b) Data Binning or Bucketing 2 c) Averaging of data 2 d) Outlier Detection 2 e) Logging	12	
Pr3	Write the programs for the following: 3 a) Perform following data processing using R 3 b) Program retrieve different attributes of data 3 e) Data pattern 3 d) Loading IP DATA ALL	18	
Pr4	Write the programs for the following: 4 a) Perform error management on the given data using pandas package 4 b) Write python/R program to create the network routing diagram from the given data on routers 4 c) Write a python/R program to build acyclic graph 4 d) Write python/R program to pick the content for BillBoards from the given data 4 e) Write a python/R program to generate GML file from given csv file 4 f) Write python/R program to plan location of warehouse from the given data 4 g) Write python/R program using data science via clustering to determine new warehouse using the data 4 h) Using the given data Write python/R program to plan the shipping routers from best-fit international logistics 4 i) Write python/R program to delete the best packing option to ship in container from the given data 4 j) Write python program to create delivery route using the given data 4 k) Write python program to crate simple forex trading planner from the given data 4 1) Write python program to process the balance sheet to ensure the only good data is processing 4 m) Write python program to generate payroll from the given data.	26	

Pr5	Write the programs for the following: Build the time hub, links and satellites.	76	
Pr6	Write the programs for the following: Transforming data	83	
Pr7	Write the programs for the following: Organizing data	qj	
Pr8	Write the programs for the following: Generating data	101	
Pr9	Write the programs for the following: Data visualisation using power Bi	116	

Practical no 1 A

Aim:-Text Delimited CSV to HORUS format

Code:-

```
import pandas as pd
sInputFileName='F:\\DS Practicals Msc-it\\Practica1 no
2\\samp1e data\\Country Code.txt'
InputData d.read csv(sInputFi1eName,encoding="1atin-1")
ProcessData=InputData
ProcessData.drop('ISO-2-CODE', axis=1,inp1ace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
ProcessData.rename(columns=('Country". 'CountryName'}, inp1ace=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(ProcessData)
OutputData=ProcessData
sOutputFileName='HORUS-CSV-Country.csv'
OutputData.to csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
```

Output:-

```
======= RESTART: F:\DS_Practicals_Msc-it\Pract
                              CountryName
CountryNumber
716                             Zimbabwe
894                               Zambia
887                                Yemen
732                        Western Sahara
876            Wallis and Futuna Islands
...                                   ...
16                         American Samoa
12                                Algeria
8                                 Albania
248                         Aland Islands
4                             Afghanistan

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

Practical no 1 B

Aim:-XML to HORUS format

Code:-

```
import pandas as pd
import xml.etree.ElementTree as ET
def xm12df(xml data):
    root = ET.XML(xml data)
    all records = []
    for i, child in enumerate(root):
        record = (}
        for subchild in child:
            record[subchild.tag] = subchild.text
        all records.append(record)
    return pd.DataFrame(all records)
sInputFileName='F:\\DS Practicals Msc-it\\Practica1 no
2\\samp1e data\\Country Code.xml'
InputData = open(sInputFi1eName).read()
ProcessDataXML=InputData
ProcessData      l2df(ProcessDataXML)
ProcessData.drop('ISO-2-CODE', axis=1,inp1ace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
ProcessData.rename(columns=('Country". 'CountryName'}, inp1ace=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)
OutputData=ProcessData
sOutputFileName='HORUS-XML-Country.csv'
OutputData.to csv(sOutputFileName, index = False)
```

Output:-

	A
1	CountryName
2	Zimbabwe
3	Zambia
4	Yemen
5	Western Sahara
6	Wallis and Futuna Islands
7	Virgin Islands, US
8	Viet Nam
9	Venezuela每(Bolivarian Republic)
10	Vanuatu
11	Uzbekistan
12	Uruguay
13	United States of America

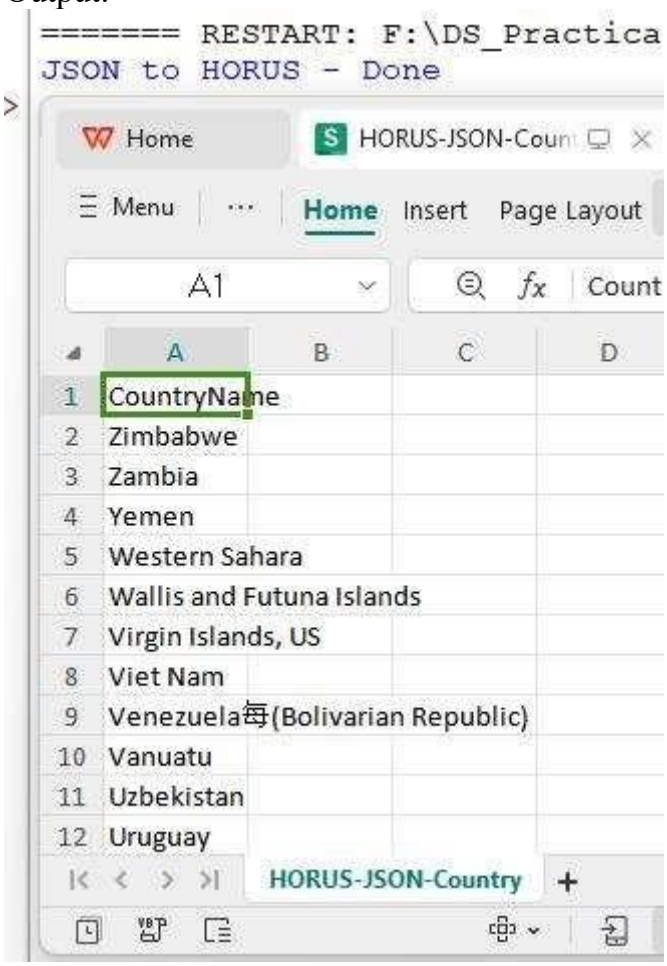
HORUS-XML-Country

Practical no 1 C
Aim:-JSON to HORUS format

Code:-

```
import pandas as pd
sInputFileName='F:\\DS Practicals Msc-it\\Practica1 no 2\\sample data\\Country Code.json'
InputData d.read son(sInputFi1eName,orient='index',encoding="latin-1")
ProcessData=InputData
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inp1ace=True)
ProcessData.rename(columns=('Country". 'CountryName'}, inp1ace=True)
ProcessData.rename(columns=('ISO-M49': 'CountryNumber'}, inp1ace=True)
ProcessData.set index('CountryNumber', inp1ace=True)
ProcessData.sort_va1ues('CountryName', axis=0, ascending=Fa1se, inplaee=True)
OutputData=ProcessData
sOutputFileName='HORUS-JSON-Country.csv'
OutputData.to csv(sOutputFileName, index = False)
print('JSON to HORUS - Done')
```

Output:-



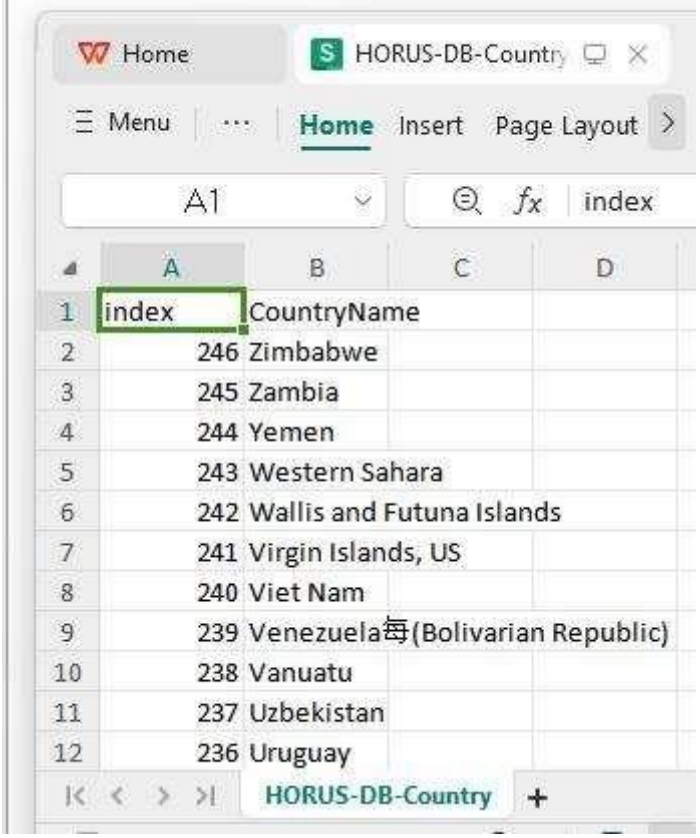
Practical no 1 D
Aim:-MySqI database to HORUS format

Code:-

```
# MySql database to HORUS format
import pandas as pd
import sqlite3 as sq
sInputFileName='F:\\DS Practicals Msc-it\\Practica1 no 2\\sample data\\utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ";'
InputData d read sql query(sSQL, conn)
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'}, inp1ace=True)
ProcessData.rename(columns=('ISO-M49': 'CountryNumber'}, inp1ace=True)
ProcessData.set index('CountryNumber', inplace=True)
ProcessData.sort_va1ues('CountryName', axis=0, ascending=Fa1se, inp1ace=True)
OutputData=ProcessData
sOutputFileName='HORUS-DB-Country.csv'
OutputData.to csv(sOutputFileName, index = False)
print('Database to HORUS - Done')
```

Output:-

```
======= RESTART: F:\DS_Practicals_
Database to HORUS - Done
```



	A	B
1	index	CountryName
2	246	Zimbabwe
3	245	Zambia
4	244	Yemen
5	243	Western Sahara
6	242	Wallis and Futuna Islands
7	241	Virgin Islands, US
8	240	Viet Nam
9	239	Venezuela每(Bolivarian Republic)
10	238	Vanuatu
11	237	Uzbekistan
12	236	Uruguay

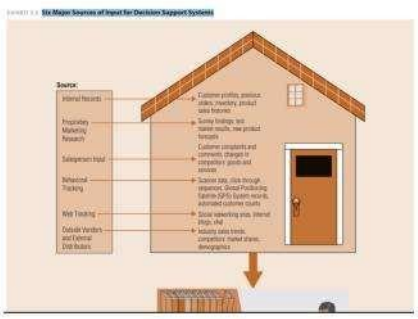
Practical no 1 E

Aim:-Picture(JPEG) to HORUS format

Code:-

```
import cv2
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
sInputFileName = 'F:\\DS_Practica1s_Msc-it\\Practica1 no 2\\sample_data\\download.png'
InputData = cv2.imread(sInputFileName, cv2.IMREAD_UNCHANGED)
if InputData is None:
    print("Error: Image not found or unable to load!")
else:
    print('Input Data Values ----------')
    print('X (Height): ', InputData.shape[0])
    print('Y (Width): ', InputData.shape[1])
    print('Channels (RGBA): ', InputData.shape[2] if len(InputData.shape) > 2 else 1)
    print('----------')
    ProcessRawData = InputData.flatten()
    y = InputData.shape[2] if len(InputData.shape) > 2 else 1
    x = int(ProcessRawData.shape[0] / y)
    ProcessData = pd.DataFrame(np.reshape(ProcessRawData, (x, y)), columns=['Red', 'Green',
'Blue', 'Alpha'][:y])
    ProcessData['XAxis'] = np.tile(np.arange(InputData.shape[1]), InputData.shape[0])
    ProcessData['YAxis'] = np.repeat(np.arange(InputData.shape[0]), InputData.shape[1])
    ProcessData = ProcessData[['XAxis', 'YAxis', 'Red', 'Green', 'Blue', 'Alpha'][:y + 2]]
    print('Rows: ', ProcessData.shape[0])
    print('Columns: ', ProcessData.shape[1])
    print('Process Data Values ----------')
    print(ProcessData.head())
    plt.imshow(cv2.cvtCo1or(InputData, cv2.COLOR_BGR2RGB))
    plt.axis('off)
    p1t.show()
    sOutputFileName = 'HORUS-Picture.csv'
    ProcessData.to_csv(sOutputFileName, index=Fa1se)
    print('Data saved to file:', sOutputFileName)
    print('Picture to HORUS - Done')
```

```
======= RESTART: F:\DS_Practicals_Msc-it\P
Input Data Values ========================
X (Height):  823
Y (Width):  1089
Channels (RGBA):  4
==========================================
Rows:  896247
Columns:  6
==========================================
Process Data Values ======================
   XAxis  YAxis  Red  Green  Blue  Alpha
0      0      0  255    255   255    255
1      1      0  255    255   255    255
2      2      0  255    255   255    255
3      3      0  255    255   255    255
4      4      0  255    255   255    255
==========================================
```



Practical no 1 F

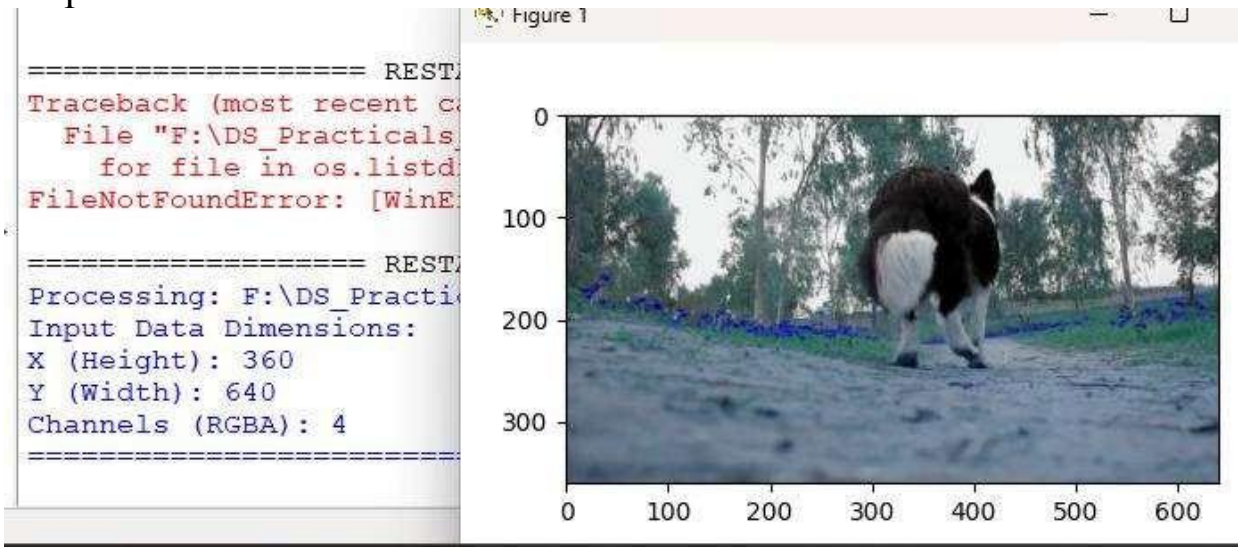
Aim:- Video to HORUS format

Code:-

```
import cv2
import pandas as pd
import matp1ot1ib.pyplot as plt
import numpy as np
import os
# Input Agreement ————————————————————————————————————————————————————————
sDataBaseDir = 'F:\\DS Practicals Msc-it\\Practica1 no 2\\videotohorus\\temp'
f= 0
# Initialize an empty list to collect the data frames
frame_data_1ist = []
# Loop through all image files in the directory
for file in os.1istdir(sDataBaseDir):
    if file.endswith(".jpg”):
        f += 1
        sInputFileName = os.path.join(sDataBaseDir, file)
        print('Processing:', sInputFileName)

       # Read the image
        InputData = cv2.imread(sInputFileName, cv2.IMREAD_UNCHANGED)
       # Convert all images to RGBA
        if len(InputData.shape) == 2:    # Grayscale
            InputData = cv2.cvtCo1or(InputData, cv2.COLOR GRAY2RGBA)
        elifInputData.shape[2] == 3:   # RGB
            InputData = cv2.cvtCo1or(InputData, cv2.COLOR_RGB2RGBA)
                # Print shape information
        print('Input Data Dimensions:')
        print('X (Height):', InputData.shape[0]) # Height
        print('Y (Width):', InputData.shape[1]) # Width
        print('Channels (RGBA):', InputData.shape[2]) # Channels (Always 4)
        print('————————————————————————————————————————————————————————————')
                # Flatten the image and create a DataFrame
        ProcessRawData = InputData.flatten()
        x = int(ProcessRawData.shape[0] / 4) # Divide by 4 channels (RGBA)
        ProcessFrameData = pd.DataFrame(np.reshape(ProcessRawData, (x, 4)))
        ProcessFrameData['FrameName'] = file    # Add the frame name
           # Append to the listframe data list.append(ProcessFrameData)
            # Display the image
        plt.imshow(cv2.cvtCo1or(InputData, cv2.COLOR_RGBA2RGB)) # Convert RGBA
to RGB for display
        p1t.show()
# Combine all frames into a single DataFrame
if f > 0:
    # Concatenate all frames into one DataFrame
    ProcessData = pd.concat(frame data list, ignore index=True)
        # Assign column names
    ProcessData.columns = ['Red', 'Green', 'Blue', 'Alpha', 'FrameName']
```

```
# Output Agreement ----------------------------------------------------
print('Processed Frames:', f)
print('Total Rows:', ProcessData.shape[0])
print('Total Columns:', ProcessData.shape[1])
    # Save the DataFrame to CSV
sOutputFileName = 'F:\\DS_Practica1s_Msc-it\\Practica1 no 2\\videotohorus\\HORUS-Movie-Frame.csv'
ProcessData.to_csv(sOutputFileName, index=Fa1se)
print('Data saved to:', sOutputFileName)
print('----------------------------------------------------------------------')
print('Movie to HORUS - Done')
print('----------------------------------------------------------------------')
```

Output:-



Practical no 1 G
Aim:- Audio to HORUS format

Code:-

```
from scipy.io import wavfile
import pandas as pd
import matp1ot1ib.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], 1abe1=sLabe1)
      plt.legend(sLegend,1oc="best")
      p1t.show()
 sInputFileName='D:\\MscIT\\DS\\DS_Practica1s\\Practicals\\Practica1 no 2\\sample_data\\2ch-
sound.wav'
print('————————————————————————————————————————————————————————')
print('Processing : ', sInputFileName)
print('————————————————————————————————————————————————————————')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData  d.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData columns=sColumns
OutputData=ProcessData
sOutputFi1eName='Practica1no 2/sample_output/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFi1eName, index = False)
   sInputFileName='D:\\MscIT\\DS\\DS_Practicals\\Practicals\\Practicalno 2\\sample_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.co1umns=sColumns
OutputData=ProcessData
sOutputFileName='Practica1 no 2/sample output/HORUS-Audio-4ch.csv'
OutputData.to csv(sOutputFileName, index = False)
sInputFileName='D:\\MscIT\\DS\\DS Practicals\\Practica1s\\Practical no 2\\samp1e data\\6ch-sound.wav'
print('═══-═══-═══-
```

Practical no 2 A
Aim:- Fixers Utilities

Code;-

```
# Fixers Utilities
import string
import datetime as dt
baddata = "       \nbad!!!            \nData Science with too many spaces is  \n"
print('Original with spaces:', '>' + baddata + '<')
cleandata = baddata.strip()
print('Cleaned:', '>' + cleandata + '<')
baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!"
cleandata = ''.join(filter(lambda x: x in string.printable, baddata))
print('Non-printable characters removed:', cleandata)
baddate = dt.date(2017, 1, 31)
formatted_baddate = format(baddate, '%Y-%m-%d')
print('Original date format:', formatted_baddate)
gooddate = dt.datetime.strptime(formatted_baddate, '%Y-%m-%d')
formatted_gooddate = format(gooddate, '%d %B %Y')
print('Reformatted date:', formatted_gooddate)
```

Output:-

```
================================ RESTART: F:\DS_Practicals_Msc-it\Practical no
Original with spaces: >
bad!!!
Data Science with too many spaces is
<
Cleaned: >bad!!!
Data Science with too many spaces is<
Non-printable characters removed: DataScience with funny characters is  bad!!!
Original date format: 2017-01-31
Reformatted date: 31 January 2017
>
```

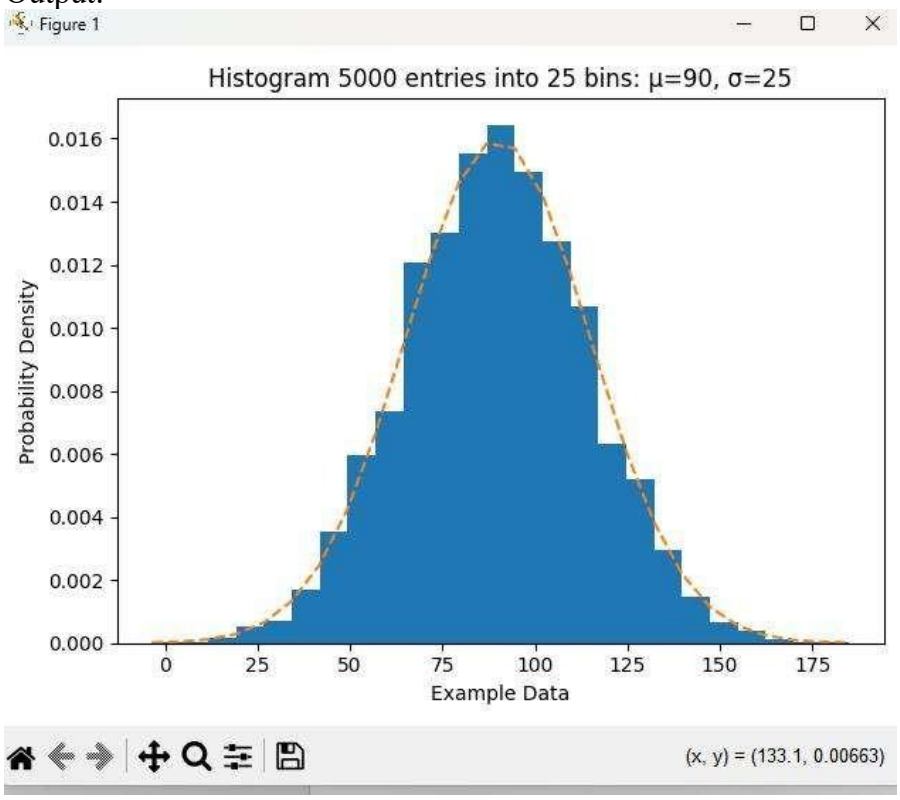
Practical no 2 B

Aim:- Data Binning or Bucketing

Code:-

```
import numpy as np
import matp1otlib.pyp1ot as plt
from scipy.stats import norm
np.random.seed(0)
mu = 90
sigma = 25
x = mu + sigma * np.random.randn(5000)
num bins = 25
fig, ax = pit.subp1ots()
n, bins, patches = ax.hist(x, num_bins, density=True)
y = norm.pdf(bins, mu, sigma)
ax.plot(bins, y, '--')
ax.set xlabe1('Example Data')
ax.set ylabe1('Probability Density')
sTitle = fHistogram (len(x)} entries into (num bins} bins: p=(mu}, o——(sigma}'
ax.set title(sTitle)
fig.tight_layout()
plt.show()
```

Output:-



Practical no 2 C

Aim:- Averaging of data

Code:-

```
# Averaging of data
import pandas as pd
sFileName='F:\\DS_Practica1s_Msc-it\\Practica1no 3\\sample_data\\IP_DATA_CORE.csv'
print('Loading :',sFi1eName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,1ow_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'],
encoding="1atin-1")
IP DATA ALL.rename(columns=('Place Name". 'Place Name'}, inplace=True)
A11Data=IP DATA ALL[['Country', 'Place Name','Latitude']]
print(A11Data)
MeanData=AllData.groupby(['Country', 'Place Name'])['Latitude'].mean()
print(MeanData)
```

OutPut:-

```
================================ RESTART: F:\DS_Practicals_Msc-it\Practical no
Loading : F:\DS_Practicals_Msc-it\Practical no 3\sample_data\IP_DATA_CORE.csv
     Country Place_Name  Latitude
0         US   New York   40.7528
1         US   New York   40.7528
2         US   New York   40.7528
3         US   New York   40.7528
4         US   New York   40.7528
...      ...        ...       ...
3557      DE     Munich   48.0915
3558      DE     Munich   48.1833
3559      DE     Munich   48.1000
3560      DE     Munich   48.1480
3561      DE     Munich   48.1480

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

Practical no 2 D

Aim:- Outlier Detection

Code:-

```
# Outlier Detection
import pandas as pd
InputFileName = "Practical no 3\samp1e_data\IP_DATA_CORE.csv"
sFileName = InputFileName
print('Loading:', sFileName)
IP DATA ALL = pd.read csv(sFi1eName, header=0, low memo False,useco1s=['Country',
'Place Name', 'Latitude', 'Longitude'], encoding="latin-1")
IP DATA ALL.rename(columns=('Place Name". 'Place Name'}, inp1ace=True)
LondonData = IP DATA ALL.1oc[IP DATA ALL['Place Name'] == 'London']
AllData = LondonData[['Country', 'Place Name', 'Latitude']]
print('All Data')
print(A11Data)
MeanData = AllData.groupby(['Country', 'Place Name'])['Latitude'].mean()
StdData = AllData.groupby(['Country', 'P1ace_Name'])['Latitude'].std()
UpperBound = float((MeanData + StdData).i1oc[0])
LowerBound = float((MeanData - StdData).iloc[0])
print('Outliers')
print('Higher than', UpperBound)
OutliersHigher = AllData[A11Data.Latitude > UpperBound]
print(OutliersHigher)
print('Lower than', LowerBound)
OutliersLower = AllData[A11Data.Latitude < LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot = AllData[(A11Data.Latitude >= LowerBound) & (A11Data.Latitude <=
UpperBound)]
print(Out1iersNot)
```

Output:-

```
Loading: F:\DS_Practicals_Msc-it\Pra
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 51.512635507867415
```

```
3438      GB     London   51.5136
Lower than 51.506176875621264
     Country Place_Name  Latitude
1915      GB     London   51.4739
Not Outliers
     Country Place_Name  Latitude
1917      GB     London   51.5085
1918      GB     London   51.5085
1922      GB     London   51.5085
1928      GB     London   51.5085
1929      GB     London   51.5085
...      ...        ...       ...
3432      GB     London   51.5092
3433      GB     London   51.5092
3434      GB     London   51.5092
3435      GB     London   51.5092
3437      GB     London   51.5085

[1485 rows x 3 columns]
```

Practical no 2 E

Aim:-Logging

Code:-

```
# Logging
import os
import logging
import uuid
import shutil
import time
Base = 'D:/MscIT'
ssubjects = ['CloudComputing', 'DS', 'SBC', 'SCT']
ssubfolders = (
    'DS'. ['DS_Practicals/Practicals/Practical no 1', 'DS_Practicals/Practicals/Practical no 2',
'DS Practicals/Practicals/Practical no 3'],
    'SBC': ['Practicals', 'presentation'],
    'SCT“. ['Chapterwise notes', 'sct    racticals/practicals', 'webnotes']

sLevels = ['debug', 'info', 'warning', 'error']
for subject in ssubjects:
    subject_dir = os.path.join(Base, subject)
    if not os.path.exists(subject dir):
        os.makedirs(subject_dir)

    subdirs = ssubfolders.get(subject, [])
    for subdir in subdirs:
        log dir = os.path.join(subject dir, subdir, 'Logging')
        if os.path.exists(1og_dir):
            shutil.rmtree(log dir)
        time.sleep(2)
        os.makedirs(log dir, exist ok=True)
        skey = str(uuid.uuid4())
        log file = os.path.join(log dir, fLogging (skey}.log')
        print('Set up:', log file)
        logging.basicConfig(
            level=logging.DEBUG,
            format='%(asctime)s %(name)-12s %(1eve1name)-8s %(message)s',
            datefmt='%m-%d %H:%M',
            fi1ename=1og file,
            fi1emode='w'

        console = logging.StreamHand1er()
        console.setLevel(logging.INFO)
        formatter = logging.Formatter('%(name)-12s: %(leve1name)-8s %(message)s')
        console.setFormatter(formatter)
        logging.getLogger('').addHand1er(console)
        for sLevel in sLevels:
            logger = logging.getLogger(f'(subject}-(subdir}-(sLevel}")
            if sLevel -   'debug'.
                logger.debug('Logging a debugging message.')
            elif sLevel == 'info':
```

```
        logger.info('Logging an informational message.')
    elif sLevel == 'warning':
        logger.warning('Logging a warning message.')
    elif sLevel == 'error".
        logger.error('Logging an error message.')
```

Output:-

```
9et up: .D:/MscIT\Ó9\D9_Practicals/Practicals/Practical nó 1\Logging\Logging_36aeb726—dd5f—4c5f—S03c—ed5086c9fó18.log
D8'D9 Practicals/Prac icals/Practical nc i-info: INFO      Logging an informa íonal messags.
os-os_Practícalg/Praczicals/Practical no l-warning: WARNIRG  Lóggíng a warning messqge.
D9-D9_Pr'dcticals/Practicals/Practiial no 1-error: ERAOR     Lojgíng an error mzssage.
jet  up:  D:/MscITFöD\o9_Practicäls/Practicals/Practical  no  2\Logging\Logging_9'6fbfc9f—234f—44'  6d-9a43-96dbccda39c9.log
.D9-D9 Practicals/Practicals/Practical no Z-info: INFO    logging an informational message.
Ö9-D9_Practicals/Practicals/Prac ical no .Z-info: iaFÖ    logging an informational message.
D9-D9_Practicals/Practicals/Practical no 2-raming: WARWIMG  Logg'ing a raming massage.
D0—DP_Rräctitals/Rracticals/Practical no 2—raming: WARNING   Logging a raming message.
D8-D9 Practicals/Prac icäls/Practical flo O-error: ERROR      logging an error msssags.
os-Ds_Practicals/Praczicals/Practical no 2'-error: ERROR      Logging an error mêssage.
set up: D:/MsrIT\D9\D9_Practirals/PracticaLs/Präctical nÖ Ü\Logging\Logging_3Zecca51-6021-44eb-S4eO-3efaö6526776.lo%
DP—DP_Practicals/Pract'cals/Practical no 3—info: INiÖ   Icgg'ing an informational message.
D9-D9 Practicals/Practicals/Practical no Z-info: INFD    logging an inCormarional message.
D9-D5_Eracticals/Practicals/Practical no Z-iufo: 7aFÖ    logging an informational message.
D5-D9_Practicals/Practicals/Practical no 3-raming: WARWING  Logg'ing a raming massage.
D9-DP_Präcticals/Pratticäls/Practical no 8-#aming: WARNING  Logging a raming message.
D8-D9 Practicäls/Prac icals/Practical nc 3-raming: WAANING  Logging a raming message.
D9-Ds_Practicalq/graczicals Practi'cal jo 3-erCor: ERROR   Dog@ing an error message.
D8-D9_Präcticals/Practicals/Practiial no 3-error: E  OR     Logging an error msssago.
DP-ÖP_Practicals/Pract'cals*Practical no 3-error: EAROR     Logging an orror message.
9st up: D:/MscIT\9%C\Practicals\Logging\Loggïng Öla6c2%%-9297-)cb7-978f-BI42?746B2?9.log
.5BC-Pxac icals-info: INFO      logging an informational message.
.9BC-Practicals-info: INFÖ   Logging an in'formational message.
DBC-Practicals-info:  INFO   Logging an informatiouäl message.
'9BC-Piacticais=in€o:  INFO   Iog'ging an infoimätional message.
DBC-'Piacticals-Warming: wAR$iNG  Logging a raming Éessaje.
9BC-Practicals-'Warniwj: BARRINg  LoggiRj q Waming /essago.
BBC-Practicals-warn}ng: WëANING  Logging a raming message.
9BC-PxacticaLs-War.ning: WARflIMG  logging a warning message.
.5BC-Practicals-error: ERROR     Logging an error message.
9BC—Practicals—error:  EARÖR     Logging an error message.                                                    .j
PBC-Practicals-error: ERROR     Logging an error message:
```

Practical no 3 A

Aim:- Perform following data processing using R

Code:-

```
library(readr)
library(tibb1e)
library(data.table)
library(e1071)
FileName  aste0('F:/DS Practicals Msc-it/Practical no 2/sample data/IP DATA CORE.csv')

IP DATA ALL <- read csv(FileName, col types = cols(ID = col integer(),Country = col character(), 'First IP Number' = col integer(), 'Last IP Number' = col integer(),Latitude = col double(), Longitude = co1_doub1e(),
                                                'Place Name' = col character(),
                                                'Post Code' = co1_character()
                                              )

IP_DATA_ALL
```

```
# A tibble: 3,562 × 8
      ID Country `Place Name` `Post Code` Latitude Longitude `First IP Number` `Last IP Number`
   <int> <chr>   <chr>        <chr>          <dbl>     <dbl>            <int>           <int>
 1     1 US      New York     10017           40.8     -74.0        204276480       204276735
 2     2 US      New York     10017           40.8     -74.0        301984864       301985791
 3     3 US      New York     10017           40.8     -74.0        404678736       404679039
 4     4 US      New York     10017           40.8     -74.0        411592704       411592959
 5     5 US      New York     10017           40.8     -74.0        416784384       416784639
 6     6 US      New York     10017           40.8     -74.0        643347456       643348479
 7     7 US      New York     10017           40.8     -74.0        643738112       643738623
 8     8 US      New York     10017           40.8     -74.0        644459264       644459519
 9     9 US      New York     10017           40.8     -74.0        644663808       644664319
10    10 US      New York     10017           40.8     -74.0        645591040       645591551
# i 3,552 more rows
```

```
IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL,syntactic = TRUE,quiet = TRUE)

IP_DATA_ALL_FIX
IP_DATA_ALL_FIX
A tibble: 3,562 × 8
```

```
      ID Country Place.Name Post.Code Latitude Longitude First.IP.Number Last.IP.Number
   <int> <chr>   <chr>      <chr>        <dbl>     <dbl>          <int>         <int>
 1     1 US      New York   10017         40.8     -74.0      204276480     204276735
 2     2 US      New York   10017         40.8     -74.0      301984864     301985791
 3     3 US      New York   10017         40.8     -74.0      404678736     404679039
 4     4 US      New York   10017         40.8     -74.0      411592704     411592959
```

```
NEWFILENAME="F:/DS_Practicals_Msc-it/Practical no 4/IP_DATA_ALL_FIX.csv"
write.csv(IP_DATA_ALL_FIX,NEWFILENAME)
IP_DATA_ALL_with_ID=rowid_to_column(IP_DATA_ALL_FIX,var="RowID")
View(IP_DATA_ALL_with_ID)
```

Practicalno4a.r* × | IP_DATA_ALL_with_ID ×

Filter

	RowID	ID	Country	Place.Name	Post.Code	Latitude	Longitude	First.IP.Number	Last.IP.Number
1	1	1	US	New York	10017	40.7528	-73.9725	204276480	204276735
2	2	2	US	New York	10017	40.7528	-73.9725	301984864	301985791

```
write.csv(IP DATA ALL FIX,NEWFILENAME)
sapply(IP DATA ALL with ID,typeof)
```

```
      RowID          ID     Country  Place.Name  Post.Code    Latitude   Longitude First.IP.Number
  "integer"   "integer" "character" "character" "character"  "double"    "double"      "integer"
Last.IP.Number
     "integer"
```

```
hist count    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)
```

	RowIDCountry	Country
1	1	DE
2	2	GB
3	3	US

```
IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_with_ID,table(Country)))
View(IP_DATA_COUNTRY_FREQ)
```

Filter

	Country	N
1	DE	677
2	GB	1502
3	US	1383

```
#Longitude :

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)
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(Longitude)))
LongitudeCSVFname="F:/New folder/DSprac/LongitudeFreq.csv"
IP_DATA_Longitude_FREQ.to_csv(LongitudeCSVFname)
```

Practicalno4a.r* × hist_Longitude_with_id ×

Filter

	RowID	Longitude
1	1	-74.0203
2	2	-74.0166
3	3	-74.0150
4	4	-74.0132
5	5	-74.0131
6	6	-74.0104

```
> #minimum value
>
> min(hist_country$Country)
[1] "DE"
>
> sapply(hist_country[,'Country'],min,na.rm=TRUE)
Country
   "DE"
>
> sapply(hist_latitude_with_id[,'Latitude'],min,na.rm=TRUE)
Latitude
 40.6888
>
> sapply(hist_Longitude_with_id[,'Longitude'],min,na.rm=TRUE)
Longitude
 -74.0203
```

```
> #maximum value
>
> max(hist_country$Country)
[1] "US"
>
> sapply(hist_country[,'Country'],max,na.rm=TRUE)
Country
   "US"
>
> sapply(hist_latitude_with_id[,'Latitude'],max,na.rm=TRUE)
Latitude
 51.5895
>
> sapply(hist_Longitude_with_id[,'Longitude'],max,na.rm=TRUE)
Longitude
    11.75
>
> #mean
> sapply(hist_latitude_with_id[,'Latitude'],mean,na.rm=TRUE)
Latitude
45.23386
>
> sapply(hist_Longitude_with_id[,'Longitude'],mean,na.rm=TRUE)
Longitude
-26.43165
>
> #median :
> sapply(hist_latitude_with_id[,'Latitude'],median,na.rm=TRUE)
Latitude
 48.1098
```

```
#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_Longitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,table(Longitude)))
setorder(IP_DATA_Longitude_FREQ-N)
IP_DATA_Longitude_FREQ[1,'Longitude']
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']
```

```
> #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']
   Country
    <char>
1:      GB
>
> IP_DATA_Longitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,table(Longitude)))
> setorder(IP_DATA_Longitude_FREQ,-N)
> IP_DATA_Longitude_FREQ[1,'Longitude']
   Longitude
      <char>
1:   -0.0955
>
> 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']
   Latitude
     <char>
1:  51.5092
```

#Range :

sapply(hist_latitude_with_id[,'Latitude'],range,na.rm=TRUE)
sapply(hist_Longitude_with_id[,'Longitude'],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 :
skewness(hist_latitude_with_id$Latitude,na.rm=FALSE,type=2)
skewness(hist_Longitude_with_id$Longitude,na.rm=FALSE,type=2)

#Missing Values :
missing_country=data.table(Country=unique(IP_DATA_ALL_with_ID[is.na(IP_DATA_ALL_with_ID['Country'])==1,]))
View(missing_country)

```
> fRange

> s apply(h1sz_1az1tu de_wi t h_1d , ' Lac1z ude ' ] , range , no. riri=TRUE}
      L az1Cude
[1,]  40.6888
[Z ,]  51. 3895

>- s Apply(h1SC_Long1z ude_wS t hot d , ' L ong1zude ' § , r ange , na. ru=TRUE)
      LDngñ nude
I+ ?  —74. 0203
[2 ,   11. 7 500

>- équarzl1es
>- s Apply(h1st_l at1tu de_ v1t h_1d , ' L¿LU Cude "I . q«ent le , no. rm=TRUE)
      L az1zude
0%    40.6888
Z 56  4O. 7 58g
50%   48.1098
Z5%   48.1526
1006  SL. 3B95
>- s apply(hñ SC_Long 1z ude_••I h_1cl , ' LDng nude ' § , quantl J e , nzt. r TRUE}
      L ong1zude
OF    —74. 0203
2 5&  —7 3. 97 78
5O%   -O. 0912
ZSX   ii.ssgs
1006  11. 7300
> 4standar d oex'1 at 1 on :

» sapply(hist_1atitude_with_id[, 'Latitude'],sd,na.rm-TRUE)
Lat1tude
4. 11052g
» s appl y(h1St_Long1 t ude_\vi th_1 d[ , ' L ongl tude ' ] ,sd , na, rri=TRUE)
Longitude
 41.54072

> #skea'ness
» skewness(hist_latizude_wizh_id#Latitude,na.rm-rALSe,zype-2)
[1]  -0. 03768831
›    skewness(hiSt_Longitude_with_id5Longitude,na.rm=rALSE,type=2)
[1]  -O. 2577 S 29

> ii ssi ng va1ues :

> ml ssi ng_count ry=d at a. t able (count ry=un1que (IP_DATA_AL L_wi t h_lo ¿1 s . na(IP_DATA_ALL_Wit h.
> v1ew(mi ss1ng_count ry)
```

Practical no 3 B

Aim:- Program retrieve different attributes of data

Code:-

```
# Program retrieve different attributes of data
import pandas as pd
sFileName='F:\DS_Practica1s_Msc-it\DS_Practica1s_Msc-it\\Practical no
4\\IP_DATA_ALL.csv'
print('Loading :',sFi1eName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,1ow_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(0,1en(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(0,1en(IP_DATA_ALL.columns)):
    cNameO1d=IP_DATA_ALL_FIX.columns[i] + '
    cNameNew=cNameO1d.strip().replace(" ", ".")
    IP_DATA_ALL_FIX.columns.values[i] = eNameNew
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print('Fixed Data Set with ID')
IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
IP_DATA_ALL_with_ID.index.names = ['RowID']
sFi1eName2='Practica1 no 4\\Retrieve_IP_DATA1.csv'
IP_DATA_ALL_with_ID.to_csv(sFi1eName2, index = True, encoding="1atin-1")
print('Done')
```

Output:-

```
======= RESTART: F:\DS_Practicals_Msc-it\Practical no 4\Practicalno4b.py =
Loading : F:\DS_Practicals_Msc-it\Practical no 4\IP_DATA_ALL.csv
Rows: 1247502
Columns: 9
Raw Data Set
Unnamed: 0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set
Unnamed:.0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
```

Practical no 3 C
Aim:-Data Pattern

Code:-

```
# Data pattern :
library(readr)
library(data.table)
FileName=paste0('F:\\DS Practicals Msc-it\\Practical no 4\\IP DATA ALL.csv')
IP_DATA_ALL <- read_csv(FileName)
hist country=data.table(Country=unique(IP DATA ALL$Country))
pattern country=data.table(Country=hist country$Country,PatternCountry=hist country$Count

oldchar=c(letters,LETTERS)
newchar=replicate(length(oldchar),"A")
for (r in seq(nrow(pattern country)))(
   s=pattern_country[r,]$PatternCountry;
   for (c in seq(length(oldchar)))(
      s=chartr(oldchar[c],newchar[c],s)

   for (n in seq(0,9,1))(
      s=chartr(as.character(n),"N",s)

   s=chartr(" ",”b",s)
   s=chartr(".",”u",s)
   pattern country[r,]$PattemCountry=s;

View(pattern country)
```

Output:-

```
> FileName=paste0('F:\\DS_Practicals_Msc-it\\Practical no 4\\IP_DATA_ALL.csv')
> IP_DATA_ALL <- read_csv(FileName)
New names:
• `` -> `...1`
Rows: 1247502 Columns: 9
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (3): Country, Place.Name, Post.Code
dbl (6): ...1, ID, Latitude, Longitude, First.IP.Number, Last.IP.Number

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Practicalno4a.r × | Practicalno4c.r* × | pattern_country

Filter

	Country	PatternCountry
1	BW	AA
2	NE	AA
3	MZ	AA

Practical no 3 D

Aim:-Loading IP_DATA_ALL

Code:-

```
#   Loading IP_DATA_ALL :
import pandas as pd
sFileName='F:\\DS_Practica1s_Msc-it\\Practica1no 4\\IP_DATA_ALL.csv'
print('Loading :',sFi1eName)
IP DATA ALL=pd.read csv(sFileName,headem0,low memo   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(0,1en(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(0,1en(IP DATA ALL.columns)):
    cNameO1d=IP DATA ALL FIX.columns[i] + '
    cNameNew=cNameO1d.strip().replace(" ", ".")
    IP DATA ALL FIX.columns.values[i] = cNameNew
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print('Fixed Data Set with ID')
IP_DATA ALL_with_ID=IP_DATA ALL FIX
IP DATA ALL_with ID.index.names = ['RowID']
sFileName2='F:\\DS Practicals Msc-it\\Practica1 no 4\\Retrieve IP DATA.csv'
IP DATA ALL_with ID.to csv(sFi1eName2, index = True, encoding="1atin-1")
print('Done')
```

Output:-

```
======= RESTART: F:\DS_Practicals_Msc-it\Practical no 4\Practical
Loading : F:\DS_Practicals_Msc-it\Practical no 4\IP_DATA_ALL.csv
Rows: 1247502
Columns: 9
Raw Data Set
Unnamed: 0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set
Unnamed:.0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
Done
```

Practical no 4 A

Aim:- Perform error management on the given data using pandas package

1) Drop the Columns Where All Elements Are Missing Values

Code:-

```
import pandas as pd
sInputFileName='F:\\DS Practicals Msc-it\\Practical no 5\\practicalno5A\\Good-or-Bad.csv'
sOutputFileName='F:\\DS_Practicals_Msc-it\\Practica1 no 5\\practica1no5A\\Good-or-Bad-01.csv'

### Import Warehouse

sFi1eName=sInputFileName
print('Loading :',sFi1eName)
RawData=pd.read_csv(sFi1eName,header=0)
print('## Raw Data Values')
print('##################################')
print(RawData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('##################################')

sFi1eName=sInputFileName
RawData.to_csv(sFi1eName, index = False)

TestData=RawData.dropna(axis=1, how='all')

print('##################################')
print('## Test Data Values')
print('##################################')
print(TestData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',TestData.shape[0])
print('Co1umns :',TestData.shape[1])
print('##################################')

sFi1eName=sOutputFi1eName
TestData.to_csv(sFileName, index = False)

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

Output:-

```
= RËSTÄĘJ:. F:\D5_Pzactícals_Nsc-ït\Pzactïcal.no 5\pzactïcalno5A\pzactícalno5a.py
Loading : F:\DS_Practicals_Msc-it\PzcctiCal nD E\pIGCtiCalaGSA\CDod-oz-3ad.csy

## Raw Data Values

       ID FieìdA  Fieìd0,FieldC  FieldD  FieldE   FieldF  FieldG
0     1.0   Good                 1024.0     NaN  10241.0       1
                     N N    i5   512.0     NaN   5121.0       2
                   C!  C!       256.0     NaN    256.0       3
'3    '  ü   Cood                 NaN     NaN    211.0       4
                  bei:N      aN   64.0     NaN   6411.0       5
                                  32.0     NaN     32.0       6
6     ''0          Better   Best  16.0     NaN   1611.0       7
                *     NaN   2est   8.0     NaN   8111.0       ·
      SEO    NaA     NaN    NaN   4.0     NaN     41.0       9
                       B      C   2.0     NaN  21111.0      10
lO    NaN    NaN     NaN    NaN   NaN     NaN      NaN      11
11   10.0   Good   Better   Best 1024.0    NaN 102411.0      12
12   lO.O   Cood      NaN   Best  512.0    NaN    512.0      13
13   lO.O   Cood   Better    NaN  256.0    NaN   1256.0      1.1
ì4   l0.ü   Good   Better   Best   NaN     NaN      NaN      15
IG   IO.U   Good   Better    NaN   64.0    NaN    164.0      16
16   lO.O   Good      NaN   Best   32.0    NaN    322.0      17''
17'  lO.O    NaN   Better   Best   16.0    NaN    163.0      1°'
18   lO.O    NaN     NaN   2est    8.0    NaN    844.0      19
19   lO.O    NaN     NaN    NaN    4.0    NaN   4555.0      ZO
ZO   lO.O     'Ä      B      C     2.0    NaN    111.0      'l.

##-Dàta Frofile

Rows : ṣI
Columns   8
####################
####################
## Test Data Values

      ID FieldA  FîeldB Fields  FieldD    FieldF  FieldG
1    1.1   Good  Beuzei   Bess  1124.1   lJ241.1       1
î    2.1   Good     NaN   Bess   512.1    5121.1       2
1    3.1   Good  Bearer    NaN   25f.1                 3
3    9.1   Good  Beucei   Best     NaN     2 1.1       9
4    5.1   Good  Beucei    NaN    f4.1    f9ll.1       5
5    6.0   Good     NaN   Best    32.0      32.0       6
6    7.0    NaN  Better   Best    16.0    1611.0       7
7    8.0    NaN     NaN   Best     8.0    8111.0       8
8    9.1    NaN     NaN    NaN     4.1      41.1       9

1Ct  NaN    NaN     NaN    NaN     NaN       NaN      11
îl   lD.1   Good   i5e rev  Bess  102.4.1  lD29ll.1    12
12   lD.1   Good     ŒaO   Bess   B.l2.1    B.lZ.1     13
13   10.0   Good  Better    NaN   256.0               14
14   10.0   Good  Better   Best     NaN       NaN      15
15   10.0   Good  Better    NaN    64.0     164.0      16
16   10.0   Good     NaN   Best    32.0     322.0      17
17   10.0    NaN  Better   Best    16.0     163.0      18
18   10.0    NaN     NaN   Best     8.0     844.0      19
19   10.0    NaN     NaN    NaN     4.0    4sss.o      20

## Daoş Profile

Rows : 21


### Done!! ####################
```

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

Code:-

```
import pandas as pd
sInputFileName='F:\\DS Practicals Msc-it\\Practical no 5\\practicalno5A\\Good-or-Bad.csv'
sOutputFileName='F:\\DS Practicals Msc-it\\Practica1 no 5\\practica1no5A\\Good-or-Bad-01.csv'
sFi1eName=sInputFileName
print('Loading :',sFi1eName)
RawData=pd.read csv(sFileName,headem0)
print(RawData)
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
sFi1eName=sInputFileName
RawData.to csv(sFileName, index = False)
TestData=RawData.dropna(axis=1, how='any')
print('Test Data Values')
print(TestData)
print('Data Profile')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
sFileName=sOutputFileName
TestData.to_csv(sFileName, index = False)
print('Done')
```

Output:-

```
===================
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5A\Good-or-Bad.cs
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
Rows : 21
Columns : 8
```

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

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

Code:-

```
import pandas as pd
sInputFileName='F:\\DS Practicals Msc-it\\Practica1 no 5\\practicalno5A\\Good-or-Bad.csv'
sOutputFileName='F:\\DS Practicals Msc-it\\Practica1 no 5\\practica1no5A\\Good-or-Bad-01.csv'
sFi1eName=sInputFileName
print('Loading :',sFi1eName)
RawData=pd.read csv(sFileName,headem0)
print('Raw Data Values')
print(RawData)
print('Data Profile')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
sFi1eName=sInputFileName
RawData.to csv(sFi1eName, index = False)
TestData=RawData.dropna(thresh=2)
print('## Test Data Values')
print(TestData)
print('## Data Profile')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
sFi1eName=sOutputFileName
TestData.to csv(sFileName, index = False)
print('Done')
```

Output:-

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

Practical no 4 B

Aim:-4 Write python/R program to create the network routing diagram from the given data on routers

1) Assess-Network-Routing-Customer

Code:-

```
import pandas as pd
pd.options.mode.chained assignment = None
sInputFileName='F:\\DS Practicals Msc-it\\Practical no 5\\practicalno5B\\Assess-Network-Routing-Customer.csv'
sOutputFileName='F:\\DS Practicals Msc-it\\Practica1 no 5\\practica1no5B\\Assess-Network-Routing-Customer.gm1'
sFileName=sInputFileName
print('Loading :',sFi1eName)
CustomerData d.read csv(sFileName,headem0,low memory=False, encoding="latin-1")
print('Loaded Country:',CustomerData.columns.values)
print(CustomerData.head())
print('Done')
```

Output;-

```
================ RESTART: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Practicalno5b_ Assess-N
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Assess-Network-Routing-Customer.csv
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
  Customer_Country_Code  ... Customer_Country_Name
0                    BW  ...              Botswana
1                    BW  ...              Botswana
2                    BW  ...              Botswana
3                    BW  ...              Botswana
4                    NE  ...                 Niger

[5 rows x 5 columns]
Done
```

	A	B	C	D	E	F	G
1	Customer_	Customer_	Customer_	Customer_	Customer_Country_Name		
2	BW	Gaborone	-24.6464	25.9119	Botswana		
3	BW	Francistowı	-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		

2) Assess-Network-Routing-Company

Cdoe:-

```
import pandas as pd
pd.options.mode.chained assignment = None

sInputFileNamel='F:\\DS_Practica1s_Msc-it\\Practica1 no
5\\practicalno5B\\Retrieve Country Code.csv'
sInputFileName2='F:\\DS_Practica1s_Msc-it\\Practica1 no
5\\practica1no5B\\Retrieve Router Location.csv'
sInputFileName3='F:\\DS Practicals Msc-it\\Practical no 5\\practicalno5B\\Retrieve IP DATA
(1)  '

sOutputFileName='F:\\DS_Practica1s_Msc-it\\Practical no 5\\practica1no5B\\Assess-Network-
Routing-Company.csv'
sFi1eName=sInputFileName1
print('Loading :',sFi1eName)
CountryData  d.read csv(sFileName,headem0,low memo  False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
print('Changed :',CountryData.columns.values)
CountryData.rename(columns=('Country". 'Country_Name'}, inp1ace=True)
CountryData.rename(columns=('ISO-2-CODE“. 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inp1ace=True)
CountryData.drop('ISO-3-Code', axis=1, inplace=True)
CountryData.drop('RowID', axis=1, inp1ace=True)
print('To :',CountryData.columns.values)
sFi1eName=sInputFi1eName2
 print('Loading :',sFi1eName)
 CompanyData  d.read csv(sFileName,headem0,low memo  False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('Changed :',CompanyData.columns.values)
CompanyData.rename(columns=('Country': 'Country_Code'}, inplace=True)
print('To :',CompanyData.columns.values)
sFileName=sInputFi1eName3
print('Loading :',sFi1eName)
CustomerRawData=pd.read csv(sFileName,header=0,low memory=Fa1se, encoding="1atin-1")
print('Loaded Customer :',CustomerRawData.columns.values)
CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('Remove Blank Country Code')
print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])
print('Changed :',CustomerData.columns.values)
CustomerData.rename(columns=('Country': 'Country_Code'}, inplace=True)
print('To  :',CustomerData.columns.values)
print('Merge Company and Country Data')
CompanyNetworkData=pd.merge(
        CompanyData,
        CountryData,
        how='inner',
        on='Country_Code'        )
print('Change ',CompanyNetworkData.columns.values)
```

```
for i in CompanyNetworkData.columns.values:
    j='Company '+i
    CompanyNetworkData.rename(co1umns={i: j}, inplace=True)
print('To ', CompanyNetworkData.columns.values)
sFi1eName=sOutputFi1eName
print('Storing :', sFileName)
CompanyNetworkData.to_csv(sFileName, index = False, encoding="latin-1")
print('Done!!')
```

Output:-

```
================= RESTART: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Practicalno5b_Assess
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Retrieve_Country_Code.csv
Loaded Country: ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
Changed : ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
To : ['Country_Name' 'Country_Code']
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Retrieve_Router_Location.csv
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude']
Changed : ['Country' 'Place_Name' 'Latitude' 'Longitude']
To : ['Country_Code' 'Place_Name' 'Latitude' 'Longitude']
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Retrieve_IP_DATA (1).csv
Loaded Customer : ['RowID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude']
Remove Blank Country Code
Reduce Rows from 1247502  to  1150323
Changed : ['RowID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude']
To : ['RowID' 'Country_Code' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude']
Merge Company and Country Data
Change  ['Country_Code' 'Place_Name' 'Latitude' 'Longitude' 'Country_Name']
To  ['Company_Country_Code' 'Company_Place_Name' 'Company_Latitude'
 'Company_Longitude' 'Company_Country_Name']
Storing : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Assess-Network-Routing-Company.csv
Done!!
```

	A	B	C	D	E	F
1	Company_C	Company_F	Company_L	Company_L	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	40.7627	-73.9727	United States of America	

3) Assess-Network-Routing-Node

Code:-

```
import pandas as pd
pd.options.mode.chained assignment = None
sInputFileName='F:\\DS Practicals Msc-it\\Practica1 no 5\practicalno5B\Retrieve_IP_DATA.csv'
sOutputFileName='F:\\DS Practicals Msc-it\\Practical no 5\practicalno5B\Assess-Network-Routing-Node.csv'
sFi1eName=sInputFileName
print('Loading :',sFileName)
IPData=pd.read csv(sFi1eName,header=0,low memory=Fa1se, encoding="latin-1")
print('Loaded IP :', IPData.columns.values)
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'}, inp1ace=True)
IPData.rename(columns=('Place.Name': 'Place Name'}, inp1ace=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'}, inp1ace=True)
print('To :',IPData.columns.values)
print('Change ',IPData.columns.values)
for i in IPData.columns.values:
    j='Node '+i
    IPData.rename(columns=(i: j}, inp1ace=True)
print('To ', IPData.columns.values)
sFi1eName=sOutputFi1eName
print('Storing :', sFileName)
IPData.to_csv(sFileName, index = False, encoding="1atin-1")
print('Done')
```

Output:-

```
================== RESTART: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Practicalno5b_As
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Retrieve_IP_DATA.csv
Loaded IP : ['RowID' 'ID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude'
 'First.IP.Number' 'Last.IP.Number']
Changed : ['RowID' 'ID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude'
 'First.IP.Number' 'Last.IP.Number']
To : ['Country_Code' 'Place_Name' 'Post_Code' 'Latitude' 'Longitude'
 'First_IP_Number' 'Last_IP_Number']
Change  ['Country_Code' 'Place_Name' 'Post_Code' 'Latitude' 'Longitude'
 'First_IP_Number' 'Last_IP_Number']
To  ['Node_Country_Code' 'Node_Place_Name' 'Node_Post_Code' 'Node_Latitude'
 'Node_Longitude' 'Node_First_IP_Number' 'Node_Last_IP_Number']
Storing : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5B\Assess-Network-Routing-Node.csv
Done
```

	A	B	C	D	E	F	G	H
1	Node_Cou	Node_Plac	Node_Post	Node_Latit	Node_Long	Node_First	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	

Practical no 4C

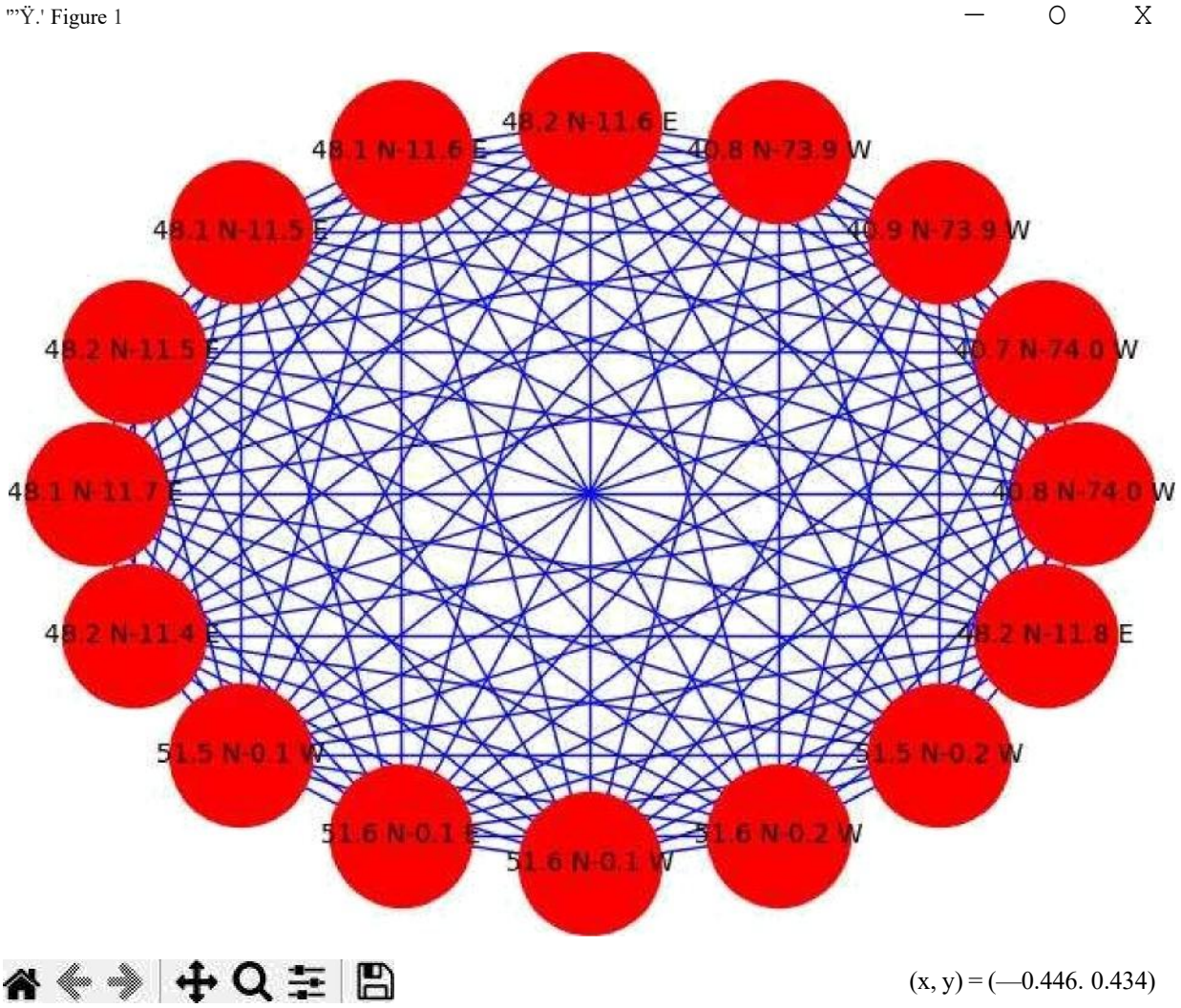
AIm:-Write a Python / R program to build directed acyclic graph.

1) Assess_DAG_GPS

Code:-

```
import networkx as nx
import matp1ot1ib.pyplot as plt
import pandas as pd
import os
sInputFileName 'F:\\DS Practicals Msc-it\\Practical no 5\\Practica1no5C\\Retrieve Router Location.csv'
sOutputFileName 'F:\\DS Practicals Msc-it\\Practical no 5\\Practicalno5C\\Assess-DAG-Company-GPS.png'
if not os.path.exists(sInputFileName):
    print(f'Error: Input file '(sInputFileName}' not found!")
    exit()
print(fi'Loading data from: (sInputFileName}")
CompanyData = pd.read csv(sInputFi1eName, header0, encoding="latin-1")
required columns = ['Latitude', 'Longitude']
if not all(col in CompanyData.columns for col in required columns):
    print(f'Error: Input file must contain the columns: (required_columns}")
    exit()
CompanyData['Latitude'] = CompanyData['Latitude'].round(1)
CompanyData['Longitude'] = CompanyData['Longitude'].round(1)

def format coord(lat, lon):
    lat dir = 'S' if lat < 0 else 'N'
    1on_dir = 'W' if lon < 0 else 'E'
    return I'{abs(lat)} {lat dir}-(abs(lon)} (lon dir}"
CompanyData['GPS'] CompanyData.apply(lambda row: format_coord(row['Latitude'], row['Longitude']), axis=1)
G = nx.Graph()
nodes = CompanyData['GPS'].unique()
G.add nodes from(nodes)
for i, nodel in enumerate(nodes):
    for node2 in nodes[i + 1:]:
        G.add_edge(nodel, node2)
print(fi'Nodes: (1en(G.nodes())}")
print(fi'Edges: (1en(G.edges())}")
print(f"Saving graph to: (sOutputFileName}")
pos = nx.circular layout(G)
nx.draw(G, pos,
        node co1om'red', edge co1or='b1ue',
        with_1abels=True, node_size=3000, font size=10)
plt.savefig(sOutputFileName)
plt show()Output:-
```

```
Loading data from: F:\DS_Prac
_Router_Location.csv
Nodes: 16
Edges: 120
Saving graph to: F:\DS_Practi
-Company-GPS.png
```


"'Ÿ.' Figure 1
48.2 N-11.6 E
40.8 N-73.9 W
48.1 N-11.6 E
40.9 N-73.9 W
48.1 N-11.5 E
40.7 N-74.0 W
48.2 N-11.5 E
40.8 N-74.0 W
48.1 N-11.7 E
48.2 N-11.8 E
48.2 N-11.4 E
51.5 N-0.2 W
51.5 N-0.1 W
51.6 N-0.2 W
51.6 N-0.1 E
51.6 N-0.1 W
(x, y) = (—0.446. 0.434)

2) Practicalno5c_Assess_DAG_Location

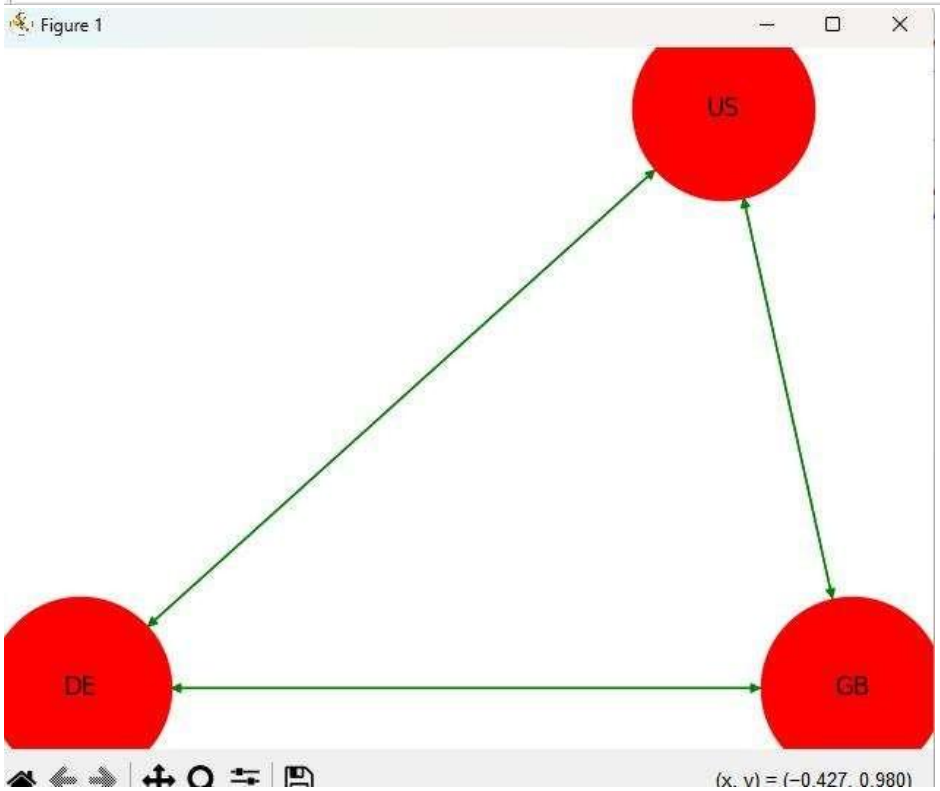
Code:-

```
import networkx as nx
import matp1ot1ib.pyp1ot as plt
import pandas as pd
sInputFileName='F:\\DS Practicals Msc-it\\Practica1 no
5\\Practica1no5C\\Retrieve_Router_Location.csv'
sOutputFileNamel='F:\\DS Practicals Msc-it\\Practical no 5\\Practica1no5C\\Assess-DAG-
Company-Country.png'
sOutputFileName2='F:\\DS Practicals Msc-it\\Practical no 5\\Practica1no5C\\Assess-DAG-
Company-Country-Place.png'
sFi1eName=sInputFileName
print('Loading :',sFi1eName)
CompanyData d.read_csv(sFileName,header=0,1ow_memory=False, encoding="1atin-1")
print('Loaded Company :',CompanyData.columns.values)
print(CompanyData)
print('Rows : ',CompanyData.shape[0])
G1=nx.DiGraph()
G2=nx.DiGraph()
for i in range(CompanyData.shape[0]):
    G1.add_node(CompanyData['Country'][i])
    sPlaceName= CompanyData['Place Name'][i] + '-' + CompanyData['Country'][i]
    G2.add_node(sPlaceName)
for n1 in GI.nodes():
    for n2 in G1.nodes():
        ifnl != n2:
            print('Link :',n1,' to ', n2)
            G1.add_edge(n1,n2)
print("Nodes of graph: ")
print(GI.nodes())
print("Edges of graph: ")
print(G1.edges())
sFi1eName=sOutputFi1eName1
 print('Storing :', sFileName)
 nx.draw(G1,pos=nx.spectra1_1ayout(G1),
        node colom'r',edge_colom'g',
        with_1abels=True,node_size=8000,
        font size=12)
plt.savefig(sFileName)
p1t.show()
 for n1 in G2.nodes():
    for n2 in G2.nodes():
        ifnl != n2:
            print('Link :',n1,' to ', n2)
            G2.add edge(n1,n2)
print("Nodes of graph: ")
print(G2.nodes())
print(”Edges of graph: ")
print(G2.edges())
 sFi1eName=sOutputFi1eName2
```

```
print('Storing :', sFileName)
nx.draw(G2,pos=nx.spectral layout(G2),
        node color='r',edge color='b',
        with labels=True,node size=8000,
        font size=12)
plt.savefig(sFileName)
plt.show()
```

OUtput:-

```
= RESTART: F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5C\Practicalno5c_Assess_DAG_Location.py
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5C\Retrieve_Router_Location.csv
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude']
    Country Place_Name  Latitude  Longitude
0        US   New York   40.7528   -73.9725
1        US   New York   40.7214   -74.0052
2        US   New York   40.7662   -73.9862
3        US   New York   40.7449   -73.9782
4        US   New York   40.7605   -73.9933
..      ...        ...       ...        ...
145      DE     Munich   48.1475    11.4635
146      DE     Munich   48.0915    11.5392
147      DE     Munich   48.1833    11.7500
148      DE     Munich   48.1000    11.4667
149      DE     Munich   48.1480    11.7434

[150 rows x 4 columns]
Rows :  150
Link : US  to  DE
Link : US  to  GB
Link : DE  to  US
Link : DE  to  GB
Link : GB  to  US
Link : GB  to  DE
Nodes of graph:
['US', 'DE', 'GB']
Edges of graph:
[('US', 'DE'), ('US', 'GB'), ('DE', 'US'), ('DE', 'GB'), ('GB', 'US'), ('GB', 'DE')]
Storing : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5C\Assess-DAG-Company-Country.png
```



Practical no 4 D

Aim:-Write a Python / R **program to pick the content for Bill Boards from** the given data.

Code:-

```
import sqlite3 as sq
import pandas as pd
sInputFileNamel='F:\\DS_Practica1s_Msc-it\\Practica1 no
5\\Practicalno5D\\Retrieve DE Billboard Locations.csv'
sInputFileName2='F:\\DS Practicals Msc-it\\Practical no
5\\Practica1no5D\\Retrieve Online Visitor.csv'
sOutputFileName='F:\\DS Practicals Msc-it\\Practica1 no 5\\Practica1no5D\\Assess-DE-
Billboard-Visitor.csv'
sDatabaseName='F:\\DS Practicals Msc-it\\Practica1 no 5\\Practicalno5D\\krennwallner.db'
conn = sq.connect(sDatabaseName)
sFi1eName=sInputFileName1
print('Loading :',sFileName)
BillboardRawData d.read csv(sFi1eName,headem0,low memory=Fa1se, encoding="latin-1")
BillboardRawData.drop duplicates(subset=None, keep='first', inp1ace=True)
Bi11boardData=Bi11boardRawData
print('Loaded Company :',BillboardData.columns.values)
sTab1e='Assess_Bi11boardData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardData.to_sql(sTable, conn, if exists="replace")
print(Bi11boardData.head())
print('Rows : ',BillboardData.shape[0])
sFi1eName=sInputFileName2
print('Loading :',sFi1eName)
VisitorRawData d.read_csv(sFi1eName,header=0,1ow_memory=False, encoding="1atin-1")
VisitorRawData.drop duplicates(subset=None, keep='first', inp1ace=True)
VisitorData=VisitorRawData[VisitorRawData.Country=='DE']
print('Loaded Company :',VisitorData.columns.values)
sTab1e='Assess VisitorData'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to sql(sTab1e, conn, if exists="replace")
print(VisitorData.head())
print('Rows : ',VisitorData.shape[0])
sTable='Assess BillboardVisitorData'
print('Loading :',sDatabaseName,' Table:',sTab1e)
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 d.read_sql query(sSQL, conn)
sTab1e='Assess BillboardVistorsData'
print('Storing :',sDatabaseName,' Table:',sTable)
Bi11boardVistorsData.to sql(sTable, conn, if exists="replace")
print(Bi11boardVistorsData.head())
print('Rows : ',BillboardVistorsData.shape[0])
print('Storing :', sFileName)
sFi1eName=sOutputFi1eName
BillboardVistorsData.to csv(sFi1eName, index = False)
print('Done')
```

Output:-

```
================= RESTART: F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5D\Practicalno5d_Picking_Content_for_Bill
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5D\Retrieve_DE_Billboard_Locations.csv
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude']
Storing : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5D\krennwallner.db  Table: Assess_BillboardData
  Country         Place_Name  Latitude  Longitude
0      DE               Lake   51.7833     8.5667
1      DE               Horb   48.4333     8.6833
2      DE         Hardenberg   51.1000     7.7333
3      DE  Horn-bad Meinberg   51.9833     8.9667
4      DE             Winkel   51.5500    13.3833
Rows :  8873
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5D\Retrieve_Online_Visitor.csv
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude' 'First_IP_Number'
 'Last_IP_Number']
Storing : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5D\krennwallner.db  Table: Assess_VisitorData
       Country Place_Name  Latitude  Longitude  First_IP_Number  Last_IP_Number
368678      DE       Lake   51.7833     8.5667       1418011904      1418012159
368679      DE       Lake   51.7833     8.5667       1528171008      1528171263
368680      DE       Lake   51.7833     8.5667       1528177920      1528178175
408664      DE       Horb   48.4333     8.6833        777246720       777246975
408665      DE       Horb   48.4333     8.6833       1339562496      1339562751
Rows :  75999
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5D\krennwallner.db  Table: Assess_BillboardVisitorData
Storing : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5D\krennwallner.db  Table: Assess_BillboardVistorsData
  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

[5 rows x 9 columns]
Rows :  181235
Storing : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5D\Retrieve_Online_Visitor.csv
Done
```

	A	B	C	D	E	F	G	H	I	J
1	BillboardCo	BillboardPl	BillboardLa	BillboardLo	VisitorCour	VisitorPlace	VisitorLatit	VisitorLong	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	

Practical no 4E

Aim:-Write a python/R program to generate GML file from given csv file

1) Understanding Your Online Visitor Data

Code:-

```
import networkx as nx
import sqlite3 as sq
import pandas as pd
from geopy.distance import geodesic

sTab1e='Assess BillboardVisitorData'
sOutputFileName='F:\\DS Practicals Msc-it\\Practica1 no 5\\Practica1no5E\\Assess-DE-Billboard-Visitor.gm1'

sDatabaseName='F:\\DS_Practica1s_Msc-it\\Practica1 no 5\\Praetica1no5E\\krennwa11ner.db'
com = 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.Bi11boardLatitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "    'N' || ROUND(ABS(A.BillboardLatitude),3)"
sSQL=sSQL+ " END ) AS sBillboardLatitude,"

sSQL=sSQL+ " (CASE WHEN A.Bi11boardLongitude < 0 THEN "
sSQL=sSQL+ "    'W' || ROUND(ABS(A.Bi11boardLongitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "    'E' || ROUND(ABS(A.Bi11boardLongitude),3)"
sSQL=sSQL+ " END ) AS sBillboardLongitude,"

sSQL=sSQL+ " A.VisitorCountry,"
sSQL=sSQL+ " A.VisitorP1aceName,"
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_Bi11boardVistorsData AS A;"
BillboardVistorsData   d.read_sql query(sSQL, conn)
print('################')


BillboardVistorsData['Distance']=Bi11boardVistorsData.apply(lambda row:
      round(
      geodesic((row['Bi11boardLatitude'],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]

      sNodel='B-'+ BillboardVistorsData['sBi11boardLatitude'][i] + '-'
      sNode1=sNode1 + BillboardVistorsData['sBi11boardLongitude'][i]
      G.add node(sNode1,
                  Nodetype='Billboard',
                  Country=Bi11boardVistorsData['Bi1lboardCountry'][i],
                  PlaceName=BillboardVistorsData['BillboardP1aceName'][i],
                  Latitude=round(BillboardVistorsData['Bi11boardLatitude'][i],3),
                  Longitude=round(BillboardVistorsData['Bi11boardLongitude'][i],3))

      sNode2='M-'+ BillboardVistorsData['sVisitorLatitude'][i] + '-'
      sNode2=sNode2  + BillboardVistorsData['sVisitorLongitude'][i]
      G.add node(sNode2,
                  Nodetype='Mobile',
                  Country=Bi11boardVistorsData['VisitorCountry'][i],
                  PlaceName=Bi11boardVistorsData['VisitorP1aceName'][i],
                  Latitude=round(BillboardVistorsData['VisitorLatitude'][i],3),
                  Longitude=round(BillboardVistorsData['VisitorLongitude'][i],3))

      print('Link Media Hub :',sNode0,' to Billboard : ', sNodel)
      G.add edge(sNode0,sNodel)

      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('##################################')
sFi1eName=sOutputFi1eName
print('##################################')
print('Storing :', sFileName)
print('##################################')
nx.write gm1(G,sFileName)
sFileName=sFileName +'.gz'
nx.write gm1(G,sFileName)


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

Output:-: This will produce a set of demonstrated values onscreen, plus a graph data file named Assess-DE-Billboard-Visitor.gml. (It takes a long time to complete the process, after completion the gm1 file can be viewed in text editor)

2) Planning an Event for Top-Ten Customers

Code:-

```
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
Company='02-Krennwa11ner'
sInputFileName='F:\\DS_Practicals_Msc-it\\Praetica1 no
5\\Practica1no5E\\Retrieve Online Visitor.csv'
sDatabaseName='F:\\DS_Practica1s_Msc-it\\Practica1 no 5\\Practica1no5E\\krennwa11ner.db'
conn = sq.connect(sDatabaseName)
sFileName=sInputFileName
print('Loading :',sFi1eName)
VisitorRawData d.read csv(sFi1eName,
                          header=0,
                          low memo False,
                          encoding="1atin-1",
                          skip blank 1ines=True)
VisitorRawData.drop duplicates(subset=None, keep='first', inp1ace=True)
VisitorData=VisitorRawData
print('Loaded Company :',VisitorData.columns.values)
sTab1e='Assess_Visitor'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print(VisitorData.head())

print('Rows : ',VisitorData.shape[0])
sView='Assess Visitor Uselt'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS" + sView + ";"
```

```
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " A Country,"
sSQL=sSQL+ " A.Place_Name,"
sSQL=sSQL+ " A.Latitude,"
sSQL=sSQL+ " A.Longitude,"
sSQL=sSQL+ " (A.Last IP Number - A.First IP Number) AS UsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess Visitor as A"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " Country is not null"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " P1ace_Name is not null;"
sql.execute(sSQL,conn)


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)


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('Loading :',sDatabaseName,' Table:',sTable)
    sSQL=" SELECT "
    sSQL=sSQL+ " *"
    sSQL=sSQL+ " FROM"
    sSQL=sSQL+ " " + sTable + ";"
    TopData   d.read sql query(sSQL, conn)

    print(TopData)


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


print('Done')
```

OUtput:-

```
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5E\krennwallner.db  Table: Assess_Tot
  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 : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5E\krennwallner.db  Table: Assess_Tot
   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
Rows :  10
Done
```

Practical no 4 F

Aim:-Write a Python / R program to plan the locations of the warehouses from the given data.

Code:-

```
import pandas as pd
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="my_custom_app1ication")
InputFileName 'F:\\DS_Practicals_Msc-it\\Practical no 5\\practicalno5F\\Retrieve GB_Postcode Warehouse.csv'
OutputFileName 'F:\\DS Practicals Msc-it\\Practica1 no 5\\practicalno5F\\Assess GB Warehouse Address.csv'
sFileName = InputFileName
print('Loading :', sFileName)
Warehouse = pd.read csv(sFileName, header0, 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, inp1ace=True)
WarehouseGoodHead.drop('id', axis=1, inp1ace=True)
WarehouseGoodHead.drop('postcode', axis=1, inp1ace=True)

WarehouseGoodTail['Warehouse Point'] = WarehouseGoodTai1.apply(
    lambda row: (str(row['latitude']) + ',' + str(row['longitude'])),
    axis=1

WarehouseGoodTai1['Warehouse Address'] = WarehouseGoodTai1.apply(
    lambda row: geolocator.reverse(row['Warehouse_Point']).address,
    axis=1
)
WarehouseGoodTail.drop('Warehouse Point', axis=1, inp1ace=True)
WarehouseGoodTail.drop('id', axis=1, inp1ace=True)
WarehouseGoodTail.drop('postcode', axis=1, inp1ace=True)
WarehouseGood = pd.concat([WarehouseGoodHead, WarehouseGoodTail], ignore index=True)
print(WarehouseGood)
sFileName = OutputFileName
WarehouseGood.to csv(sFi1eName, index=False)
print('Done')
```

Output:-

```
-=====-=-=====-=-==-= RESTART: F:\D9_Practicals_Msc-it\Practical no 5\practic
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5F\Retrieve_GB_P
     latitude  longitude                                  9arehouse_Address
 O  57.135140  -2.117310  35, Broomhill Road, Broomhill, Ashley and Broo...
1  57.138750  -2.010850  9outh Esplanade West, Tullos, Torry, Absrdesn ...
2  57.101000  -2.110600  AS2, Charlestown, Nigg, Aberdeen City, Alba / ...
3  57.108010  -2.237760  Colthill Circle, Cults, Bieldside and Milltimb...
4  57.100760  -2.270730  Johnston Gardens East, culter, Peterculter, Ab...
b 53.837717  -1.7B0013  HM Revenue and Customs, Riverside Estate, 9hip...
6  53.794470  -1.766535  Listerhills Road, Great Horton, Bradford, Best...
7  51.518556  -0.714794  Howarth Road, Stafferton may Industrial Area, ...
8  54.890523  -2.543847  Carlisle Delivery Office, Junction Street, Den...
S 57.481338  -4.223551  Railway Terrace, Longman, Crown and City Centr...
Done
```

@ Home | ds prs sanjeev.docx | @ Assess GB Warehouse Addrc x -{-

Menu | Home | Insert | Page Layout | Formulas | Data | Review | View | Tools | Smart Toolbox

E1b O. /x

•	A	B	C	D	E	F	G	H	J	K	L	M
	latitude	longitude	Warehause_Address									
2	37.13514	-2.11731	35, Broomhill Road, Broomhill, Ashley and Broomhill, Aberdeen City, Alba / Scotland, ABPO 6JA, United Kingdom									
3	37.138T3	-2.09089	South Esplanade West, Tullos, Torry, Aberdeen City, Alba/ Scotland, ABt3 9AE, United Kingdom									
4	37.101	-2.M06	A92, Charlestown, Nigg, Aberdeen City, Alba/ Scotiand, AB12 3ZT, United Kingdom									
3	37.10801	-2.23776	Colthill Circle, Cults, Bieldside and Milltimber, Milltimber, Aberdeen City, Alba/ Scotland, AB13 OEH, United Kingdom									
6	37.300T6	-2.27073	Johnston Gardens East, Culter, Peterculter, Aberdeen City, Alba/ Scotland, ABI4 OLA, United Kingdom									
T	53.B3 T	-1.780013	HM Revenue and Customs, Riverside Estate, Sfiipley, Baildon, Bradford, West Yorkshire, England, BD98 BAA, United Kingdom									
8	53.7944T	-1.766539	Listerhills Road, Great Horton, Bradford, West Yorkshire, England, BD7 1HZ, United Kingdom									
9	51.318556	-0.714794	Howarth ROad, staffertan Way Industrial Area, Fishery, Maidenhead, Windsor and Maidenhead, England, SL6 DWT, United Kingdom									
t0	54.B90923	-2.943B47	Carlisle Delivery Dffice, Junction Street, Denton Holme, Carlisle, Cumberland, England, CA1 VAA, United Kingdom									
11	57.483338	-4.223931	Railway Terrace, Longman, Crown and City Centre, Inverness, Highland, Alba / Scotland, IV99 1AA, United Kingdom									

Practical no 4 G

Aim:-Write a Python / R program using data science via clustering to determine new warehouses using the given data. .

Code:-

```
import sys
import os
import pandas as pd
InputFileName='F:\\DS_Practica1s_Msc-it\\Practica1 no
5\\practicalno5G\\Retrieve All Countries.csv'
OutputFileName='F:\\DS Practicals Msc-it\\Practica1 no
5\\practicalno5G\\Assess All Warehouse.csv'
sFi1eName=InputFi1eName
print('###########')
print('Loading :',sFileName)
Warehouse d.read_csv(sFi1eName,header=0,1ow_memory=False, encoding="latin-1")

sColumns=('XI' : 'Country',
          'X2' : 'PostCode',
          'X3' : 'PlaceName',
          'X4' : 'AreaName',
          'X5' : 'AreaCode',
          'X10' : 'Latitude',
          'X11' : 'Longitude'}
Warehouse.rename(columns=sColumns,inplace=True)
WarehouseGood=Warehouse

sFi1eName=OutputFileName
WarehouseGood.to csv(sFi1eName, index = False)
print('Done')
```

Output:-

Home | ds prs sanjeev.docx | Assess_All_Warehouse.csv

Menu | Home | Insert | Page Layout | Formulas | Dat

L7 | fx

	A	B	C	D	E	F	G	H	I
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	
7	6	AD	AD600	Sant Julià de Lòria			42.4667	1.5	
8	7	AD	AD700	Escaldes-Engordany			42.5	1.5667	
9	8	AR	3636	POZO CERC	Salta	A	-23.4933	-61.9267	
10	9	AR	4123	LAS SALADA	Salta	A	-25.7833	-64.5	
11	10	AR	4126	EL BRETE	Salta	A	-26.0667	-65.3667	
12	11	AR	4126	OVEJERO	Salta	A	-26.0833	-65.263	
13	12	AR	4126	MIRAFLORE	Salta	A	-26.0833	-65.263	
14	13	AR	4126	LA MARAVI	Salta	A	-26.0833	-65.263	
15	14	AR	4126	LA ASUNCI(	Salta	A	-26.0833	-65.263	
16	15	AR	4126	BARADERO	Salta	A	-26.0833	-65.263	
17	16	AR	4126	EL CUIBAL	Salta	A	-26.0833	-65.263	
18	17	AR	4126	CEIBAL	Salta	A	-26.1	-65.0167	
19	18	AR	4126	ALEM	Salta	A	-26.0833	-65.263	
20	19	AR	4126	EL SUNCHA	Salta	A	-26.0833	-65.263	
21	20	AR	4126	TALA	Salta	A	-26.1167	-65.2833	

Practical no 4 H

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

Code:-

```
import pandas as pd
import networkx as nx
from geopy.distance import geodesic
import sqlite3 as sq
from pandas.io import sql
InputFileName='F:\\DS Practicals Msc-it\\Practica1 no
5\\practica1no5H\\Retrieve All Countries.csv'
OutputFileName='F:\\DS Practicals Msc-it\\Praetica1 no
5\\practicalno5H\\Assess Best Logistics.gml'
sDatabaseName='F:\\DS_Practica1s_Msc-it\\Practica1 no 5\\practicalno5H\\Hillman.db'
conn = sq.connect(sDatabaseName)

sFi1eName=InputFi1eName
print('###########')
print('Loading :',sFi1eName)
Warehouse   d.read csv(sFileName,header=0,low memory=False, encoding="latin-1”)

sCo1umns=('X1' : 'Country',
          'X2' : 'PostCode',
          'X3' : 'PlaceName',
          'X4' : 'AreaName',
          'X5' : 'AreaCode',
          'X10' : 'Latitude',
          'X11' : 'Longitude'}
Warehouse.rename(columns=sColumns,inp1ace=True)
WarehouseGood=Warehouse
#print(WarehouseGood.head())

RoutePointsCountry=pd.DataFrame(WarehouseGood.groupby(['Country'])[['Latitude','Longitud
e']].mean())
#print(RoutePointsCountry.head())
print('################')
sTab1e='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('################')
sTab1e='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(RoutePointsP1aceName.head())
print('################')
sTab1e='Assess_RoutePointsP1aceName'
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('################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+  " *"
sSQL=sSQL+  " FROM"
sSQL=sSQL+ " " + sView + ";"
RouteCountries   d.read  sql  query(sSQL, conn)

RouteCountries['Distance']=RouteCountries.apply(lambda row:
      round(
      geodesic((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(
      geodesic((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  sq1_query(sSQL, conn)

RoutePlaceName['Distance']=RoutePlaceName.apply(lambda row:
      round(
      geodesic((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,sNodel)

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,sNodel)

print('Place Name:',RouteP1aceName.shape)
for i in range(RouteP1aceName.shape[0]):
    sNode0='P-' + RoutePlaceName['TargetPostCode'][i] + '-'
    sNode0=sNode0 + RoutePlaceName['TargetCountry'][i]
    G.add_node(sNode0,
               Nodetype='PostCode',
               Country=RouteP1aceName['SourceCountry'][i],
               PostCode=RouteP1aceName['TargetPostCode'][i],
               Latitude=round(RoutePlaceName['SourceLatitude'][i],4),
               Longitude=round(RouteP1aceName['SourceLongitude'][i],4))
```

```
    sNode1='L-' + RoutePlaceName['TargetPlaceName'][i] + '-'
    sNode1=sNode1 + RoutePlaceName['TargetPostCode'][i] + '-'
    sNode1=sNode1  + RoutePlaceName['TargetCountry'][i]
    G.add node(sNode1,
                Nodetype='P1aceName',
                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,sNodel)

sFi1eName=OutputFileName
print('##################################')
print('Storing :', sFileName)
print('##################################')
nx.write gm1(G,sFileName)
sFileName=sFileName +'.gz'
nx.write_gm1(G,sFileName)

print('##################################')
print('Path:', nx.shortest  ath(G,source='P-SW1-GB',target='P-01001-US',weight='distance'))
print('Path length:', nx.shortest  ath length(G,source='P-SW1-GB',target='P-01001-
US',weight='distance'))
print('Path length (1):', nx.shortest  ath length(G,source='P-SW1-GB',target='C-
GB',weight='distance'))
print('Path length (2):', nx.shortest  ath length(G,source='C-GB',target='C-
US',weight='distance'))
print('Path length (3):', nx.shortest  ath length(G,source='C-US',target='P-01001-
US',weight='distance'))
print('##################################')
print('Routes from P-SW1-GB < 2: ', nx.single source shortest  ath(G,source='P-SWl-
GB' ,cutoff=1))
print('Routes from P-01001-US < 2: ', nx.single source shortest  ath(G,source='P-01001-
US' ,cutoff=1))
print('##################################')

print('################')
print('Vacuum Database')
sSQL="VACUUM;”
sql.execute(sSQL,conn)
print('################')

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

Output:-

```
  SourceCountry SourcePostcode  ...  TargetLongitude  Distance
0            AU           0200  ...          149.1185       0.0
1            AU           0221  ...          1QS.1Q12       O.0
2            AU           O8OO  ...          130.8418       O.O
3            AU           OBO1  ...          130.B418       0.0
4            AU           O8OO  ...          130.7443       0.0

   rows
Countries: (ZO, 7)
Post code: (76Z16, 8)
Place Name: (260Z6O, lO)

Storing : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5H\Assess_Best_Logistics.gml


Path: ['P-9Kl-GB', 'C-GB', 'O-US', 'P-01001-US']
Path length: Z1Z7.452
Path length (1): 155.802
Path length (2): 90b5.869
Path length (Z): 525.781

RoGtes from P-9Bl-GB < 2:  ('P-SRl-GB': ['P—S01-GB'], 'O-GB': ['P-Rel-GB', 'C-GB'], 'L-Belgr
-Buckingham Palace-SWl-GB': ['P-SBl-GB', 'L-Buckingham Palace-EWl-GB'], 'E-Green Park-9W1-GB
y Road-sWl-GB': ['P-9Wl—GB', 'L-Horseferry Road-9Bl-GB'], 'L—Hyde Park Corner-9Wl—GB': ['P-5
ge-sWl-GB': ['P-sBl-GB', 'L-Knightsbridge-swt-GB'], 'E-Pimlico-sml-GB': ['P-swl-GB', 'L-Piml
                                                                                  -GB
                                                                                  cto
R ute» E<om P—O10 0 1—Us < 2.  ('P—010OI—Us'. ['P—O1OO1—Us ], 'c—UNS'. ['P—O1001—Us', 'c—UNS'],



Vacuum Database

Warning (from warnings module}:
  File "F:\Dfl Practicals Msc-it\Practical no 5\practicalnc0H\Practicalnc5h.py", line 27b
    sql.oxzcutc(s8QL,conn)
RutureBarning: 'pandas.io.sql.axecuto' is deprecatcd and will be rmcPad in chs future very
88####]¢##¢#  #A
### Done!! ##############################################
```

"X @ file:///"F:y'DS Practicals Msc-i+./Practical96?0no%Z05/practicaino 5H/"Assess Best Logi>4ics.gml

```
graph [
  node [
    id e
    label "C-AU"
    NOdetype "Country"
    COunl my "fl.U"
    Lat i tude f¥P . F LOAT64( - 32 . 1B91)
    Longitude NP . F LOAT64( 143 . 7176)

  node [
    id 1
    label "C-BE"
    NOde type "Country"
    COuntny "BE "
    Lat 1 tude NP . F LOAT64( 50. 64B3}
    Longitud e NP . FLDAT64(4. 5578}

  nude [
    id 2
    label "C-DE"
    ModetyQe "Country"
    Country "DE "
    L at i tude NP . FLOAT64( 51. 2126}
    Longitud e NP . F LOAT64( 16. 2662)

  nude [
    id 3
```

Practical no 4 I

Aim:-Write a Python / R program to decide the best packing option to ship in container from the given data.

Code:-

```
import pandas as pd
import sqlite3

# File paths (update these paths as needed)
InputFileName1 = "F:\\DS_Practicals_Msc-it\\Practical no 5\\practicalno5I\\Retrieve_Product.csv"
InputFileName2 = "F:\\DS_Practicals_Msc-it\\Practical no 5\\practicalno5I\\Retrieve_Box.csv"
InputFileName3 = "F:\\DS_Practicals_Msc-it\\Practical no 5\\practicalno5I\\Retrieve_Container.csv"
DatabaseFileName = "F:\\DS_Practicals_Msc-it\\Practical no 5\\practicalno5I\\hillman.db"

# SQLite connection
conn = sqlite3.connect(DatabaseFileName)

# Function to process and store data
def process_data(filename, table_name,  id_field_name, additional_filters=None, max_rows=None):
    print(f"Loading: (filename}")
    data_raw = pd.read_csv(filename, header=0, low_memory=False, encoding="latin-1")
    data_raw.drop_duplicates(inplace=True)

    # Add IDNumber dynamically
    data_raw.reset_index(drop=True, inplace=True)
    data_raw[id_field_name] = data_raw.index
    data_raw.set_index(id_field_name, inplace=True)

    # Apply filters if provided
    if additional_filters:
        for col, condition in additional_filters.items():
            data_raw = data_raw.query(f"{col) {condition}")

    # Limit the number of rows if needed
    if max_rows:
        data_raw = data_raw.head(max_rows)

    print(f"Rows in (table_name}: (data_raw.shape[0])")
    data_raw.to_sql(table_name, conn, if_exists="replace", index=True)
    print(f"Stored: (DatabaseFileName}  Table: (table_name)")
    return data_raw
# Process Product Data
ProductData = process_data(
    filename=InputFileName1,
    table_name="Assess_Product",
    id_field_name="IDNumber"
)

# Process Box Data with filtering and max row limit
BoxData = process_data(
    filename=InputFileName2,
    table_name="Assess_Box",
    id_field_name="IDNumber",
    additional_filters=("Length": "<= 1"}, # Example filter for Length
    max_rows=1000
)
```

```
# Process Container Data
ContainerData = process_data(
    filename=InputFileName3,
    table_name="Assess_Container",
    id_field_name="IDNumber"
)

# Create view for Product in Box
print("################")
print("Creating View: Assess_Product_in_Box")
sSQL = """
CREATE VIEW IF NOT EXISTS Assess_Product_in_Box AS
SELECT P.IDNumber AS ProductID, P.UnitNumber AS ProductUnit,
        B.IDNumber AS BoxID, B.UnitNumber AS BoxUnit
FROM Assess_Product AS P, Assess_Box AS B
WHERE P.ProductVolume <= B.BoxVolume;

with conn:
    conn.execute(sSQL)
print("View:  Assess_Product_in_Box created  successfully.")

# Create PalletFrame by combining data dynamically
print("################")
print("Creating PalletFrame...")
PalletFrame  =  pd.DataFrame(columns=["PalletID",  "BoxID"])
PalletID = 0

for index, box_row in BoxData.iterrows():
    if PalletFrame.empty or len(PalletFrame[PalletFrame["PalletID"] == PalletID])  >= 10:
        PalletID += 1
    PalletFrame = pd.concat([
        PalletFrame,
        pd.DataFrame(("PalletID": [PalletID], "BoxID": [index]})
    ], ignore_index=True)

# Save PalletFrame to database
PalletFrame.to_sql("Assess_Pallet", conn, if_exists="replace", index=False)
print("PalletFrame stored in database successfully.")
print("################")

# Close the database connection
conn.close()
print("Done")
```

Output;-

```
================================ RESTART: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5I\Prac
Loading: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5I\Retrieve_Product.csv
Rows in Assess_Product: 8000
Stored: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5I\hillman.db  Table: Assess_Product
Loading: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5I\Retrieve_Box.csv
Rows in Assess_Box: 1000
Stored: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5I\hillman.db  Table: Assess_Box
Loading: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5I\Retrieve_Container.csv
Rows in Assess_Container: 900
Stored: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5I\hillman.db  Table: Assess_Container
################
Creating View: Assess_Product_in_Box
View: Assess_Product_in_Box created successfully.
################
Creating PalletFrame...
PalletFrame stored in database successfully.
################
Done
```

Practical no 4 J

Aim:-Write a Python program to create a delivery route using the given data.

Code:-

```
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import networkx as nx
from geopy.distance import geodesic
nMax=3
nMaxPath=10
nSet=False
nVSet=Fa1se
InputFileNamel='F:\\DS_Practica1s_Msc-it\\Practical no
5\\practica1no5J\\Retrieve GB Postcode Warehouse.csv'
InputFileName2='F:\\DS_Practica1s_Msc-it\\Practical no
5\\practicalno5J\\Retrieve GB Postcodes Shops.csv'
OutputFileNamel='F:\\DS Practicals Msc-it\\Practical no
5\\practica1no5J\\Assess Shipping Routes.gml'
OutputFileName2='F:\\DS Practicals Msc-it\\Practica1 no
5\\practica1no5J\\Assess_Shipping_Routes.txt'
sDatabaseName='F:\\DS Practicals Msc-it\\Practica1 no 5\\practicalno5J\\hillman.db'
conn = sq.connect(sDatabaseName)
sFileName=InputFileName1
print('Loading :',sFileName)
WarehouseRawData  d.read_csv(sFi1eName,header=0,low memory=Fa1se, encoding="1atin-1")
WarehouseRawData.drop duplicates(subset=None, keep='first', inp1ace=True)
WarehouseRawData.index.name = 'lDNumber'
# 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'])

# Replace these lines
# 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'])

# With this:
frames = [WarehouseRawData.head(nMax),
          WarehouseRawData.tail(nMax),
          WarehouseRawData[WarehouseRawData.postcode == 'KA13']]
```

```
if nset:
    frames.append(WarehouseRawData[WarehouseRawData.postcode == 'SW1W'])

WarehouseData = pd.concat(frames).drop_duplicates(keep='first')

WarehouseData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Loaded Warehouses :',WarehouseData.columns.values)
sTab1e='Assess_Warehouse_UK'
print('Storing :',sDatabaseName,' Table:',sTable)
WarehouseData.to_sql(sTable, conn, if_exists="replace")
print(WarehouseData.head())
print('Rows : ',WarehouseData.shape[0])
sFi1eName=InputFileName2
print('###########')
print('Loading :',sFi1eName)
ShopRawData = d.read_csv(sFileName,header=0,low_memo = False,encoding="latin-1")
ShopRawData.drop_duplicates(subset=None, keep='first', inp1ace=True)
ShopRawData.index.name = 'IDNumber'
ShopData=ShopRawData
print('Loaded Shops :',ShopData.columns.values)
sTab1e='Assess_Shop_UK'
print('Storing :',sDatabaseName,' Table:',sTable)
ShopData.to_sql(sTable, conn, if_exists="replace")
print(ShopData.head())
print('##################################')
print('Rows : ',ShopData.shape[0])
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.1atitude,6) AS HQ_Latitude,"
sSQL=sSQL+ " round(W.1ongitude,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)
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.1atitude,6) AS Warehouse Latitude,"
sSQL=sSQL+ " round(W.1ongitude,6) AS Warehouse Longitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess Warehouse UK as W;"
sql.execute(sSQL,conn)
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()
sTable = 'Assess HQ'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData d.read sql query(sSQL, conn)
print(RouteData.head())
print('#################################')
print('HQ Rows : ',RouteData.shape[0])
for i in range(RouteData.shape[0]):
    sNode0=RouteData['H 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))
sTable = 'Assess_Warehouse'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData d.read_sq1_query(sSQL, conn)
print(RouteData.head())
```

```
print('##################################')
print('Warehouse Rows : ',RouteData.shape[0])
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 d.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))
for sNode0 in G.nodes:
    for sNodel in G.nodes:
        if G.nodes[sNode0]['Nodetype']=='HQ' and \
            G.nodes[sNode1]['Nodetype']=='HQ' and \
            sNode0 != sNodel:
                distancemeters=round(\
                                geodesics

                                G.nodes[sNode0]['Latitude'],\
                                G.nodes[sNode0]['Longitude']\
                                ),\

                                G.nodes[sNode1]['Latitude']\

                                G.nodes[sNode1]['Longitude']\

                                ).meters\
```

```
                                        ,0)
                distancemileswound(\
                                        geodesics

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

                                                  G.nodes[sNode1]['Longitude']\

                                                  ).mi1es\
                                        ,3)

                if distancemiles >= 0.05:
                      cost = round(150+(distancemiles * 2.5),6)
                      vehic1e='V001'
               else:
                      cost = round(2+(distancemiles * 0.10),6)
                      vehicle='ForkLift'

                G.add edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                            DistanceMiles=distancemi1es,
                            Cost=cost,Vehicle=vehicle)
                ifnVSet==True:
                      print('Edge-H-H:',sNode0,' to ', sNodel, \
                         ' Distance:',distancemeters,'meters',\
                         distancemiles,'miles','Cost',  cost,'Vehicle',vehicle)

        if G.nodes[sNode0]['Nodetype']=='HQ' and \
              G.nodes[sNode1]['Nodetype']=='Warehouse' and \
              sNode0 != sNodel:
                distancemeters=round(\
                                        geodesics

                                                G.nodes[sNode0]['Latitude'],\
                                                G.nodes[sNode0]['Longitude']\


                                                  G.nodes[sNode1]['Latitude']\

                                                  G.nodes[sNode1]['Longitude']\

                                                  ).meters\
                                        ,0)
                distancemiles=round(\
                                        geodesics

                                                G.nodes[sNode0]['Latitude'],\
```

```
                                                    G.nodes[sNode0]['Longitude']\


                                                     G.nodes[sNode1]['Latitude']\

                                                     G.nodes[sNode1]['Longitude']\

                                                     ).mi1es\
                                        ,3)
                if distancemiles >= 10:
                        cost = round(50+(distancemiles * 2),6)
                        vehic1e='V002'
                else:
                        cost = round(5+(distancemiles * 1.5),6)
                        vehic1e='V003'
                if distancemiles <= 50:
                        G.add edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                                        DistanceMiles=distancemi1es, \
                                        Cost=cost,Vehic1e=vehicle)
                        if nVSet==True:
                                print('Edge-H-W:',sNode0,' to ', sNodel, \
                                   ' Distance:',distancemeters,'meters',\
                                   distancemiles,'miles','Cost',  cost,'Vehic1e',vehicle)

        if nSet==True and \
                G.nodes[sNode0]['Nodetype']=='Warehouse' and \
                G.nodes[sNode1]['Nodetype']=='Warehouse' and \
                sNode0 != sNodel:
                        distancemeters=round(\
                                                geodesics

                                                    G.nodes[sNode0]['Latitude'],\
                                                    G.nodes[sNode0]['Longitude']\


                                                     G.nodes[sNode1]['Latitude']\

                                                     G.nodes[sNode1]['Longitude']\

                                                     ).meters\
                                        ,0)
                        distancemiles=round(\
                                                geodesics
                                                        (\
                                                    G.nodes[sNode0]['Latitude'],\
                                                    G.nodes[sNode0]['Longitude']\


                                                     G.nodes[sNode1]['Latitude']\
```

```
                                                G.nodes[sNode1]['Longitude']\
                                                )
                                                ).mi1es\
                                        ,3)
            if distancemiles >= 10:
                cost = round(50+(distaneemiles * 1.10),6)
                vehic1e='V004'
            else:
                cost = round(5+(distancemiles * 1.05),6)
                vehicle='V005'

            if distancemiles <= 20:
                G.add edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                            DistanceMiles=distancemi1es, \
                            Cost=cost,Vehic1e=vehicle)
                if nVSet==True:
                    print('Edge-W-W:',sNode0,' to ', sNodel, \
                      ' Distance:',distancemeters,'meters',\
                      distancemiles,'mi1es','Cost', cost,'Vehic1e',vehicle)

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

            distancemeters=round(\
                                    geodesics
                                            (\
                                            G.nodes[sNode0]['Latitude'],\
                                            G.nodes[sNode0]['Longitude']\


                                            G.nodes[sNode1]['Latitude']\

                                            G.nodes[sNode1]['Longitude']\

                                            ).meters\
                                    ,0)
            distancemiles=round(\
                                    geodesics

                                            G.nodes[sNode0]['Latitude'],\
                                            G.nodes[sNode0]['Longitude']\
                                            ),\

                                            G.nodes[sNode1]['Latitude']\

                                            G.nodes[sNode1]['Longitude']\

                                            ).mi1es\
```

```
                                        ,3)
                    if distancemiles >= 10:
                        cost = round(50+(distancemiles * 1.50),6)
                        vehic1e='V006'
                    else:
                        cost = round(5+(distancemiles * 0.75),6)
                        vehic1e='V007'
                    if distancemiles <= 10:
                        G.add edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                                   DistanceMi1es=distancemiles, \
                                   Cost=cost,Vehic1e=vehicle)
                        if nVSet==True:
                            print('Edge-W-S:',sNode0,' to ', sNodel, \
                              ' Distance:',distancemeters,'meters',\
                              distancemiles,'mi1es','Cost', cost,'Vehic1e',vehicle)

            if nSet==True and \
                G.nodes[sNode0][‘Nodetype']=='Shop' and \
                G.nodes[sNode1]['Nodetype']=='Shop' and \

G.nodes[sNode0]['WarehousePostCode']==G.nodes[sNodel]['WarehousePostCode'] and \
                sNode0 != sNodel:

                    distancemeters=round(\
                                        geodesics

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

                                                 G.nodes[sNode1]['Longitude']\

                                                 ).meters\
                                        ,0)
                    distancemiles=round(\
                                        geodesics

                                                G.nodes[sNode0]['Latitude'],\
                                                G.nodes[sNode0]['Longitude']\


                                                 G.nodes[sNode1]['Latitude']\

                                                 G.nodes[sNode1]['Longitude']\

                                                 ).mi1es\
                                        ,3)
```

```
                if distancemiles >= 0.05:
                    cost = round(5+(distancemiles * 0.5),6)
                    vehicle='V008'
                else:
                    cost = round(1+(distancemiles * 0.1),6)
                    vehic1e='V009'

                if distancemiles <= 0.075:
                    G.add edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                                DistanceMi1es=distancemiles, \
                                Cost=cost,Vehic1e=vehicle)
                    if nVSet==True:
                        print('Edge-S-S:',sNode0,' to ', sNodel, \
                          ' Distance:',distancemeters,'meters',\
                          distancemiles,'mi1es','Cost', eost,'Vehic1e',vehicle)

        if nSet==True and \
            G.nodes[sNode0][‘Nodetype']=='Shop' and \
            G.nodes[sNode1]['Nodetype']=='Shop' and \

G.nodes[sNode0]['WarehousePostCode']!=G.nodes[sNode1]['WarehousePostCode'] and \
            sNode0 != sNodel:

                distancemeters=round(\
                                geodesics

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

                                 G.nodes[sNode1]['Longitude']\

                                 ).meters\
                            ,0)
                distancemiles=round(\
                                geodesics

                                G.nodes[sNode0]['Latitude'],\
                                G.nodes[sNode0]['Longitude']\


                                 G.nodes[sNode1]['Latitude']\

                                 G.nodes[sNode1]['Longitude']\

                                 ).mi1es\
                            ,3)
```

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

                    if distancemiles <= 0.025:
                        G.add edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                                   DistanceMi1es=distaneemi1es, \
                                   Cost=cost,Vehic1e=vehicle)
                        if nVSet==True:
                            print('Edge-S-S:',sNode0,' to ', sNodel, \
                              ' Distance:',distancemeters,'meters',\
                              distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

sFi1eName=OutputFileName1
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.write gml(G,sFileName)
sFi1eName=sFi1eName +'.gz'
nx.write gm1(G,sFi1eName)
print('Nodes:',nx.number_of_nodes(G))
print('Edges:',nx.number of edges(G))
sFi1eName=OutputFileName2
print('#################################')
print('Storing :', sFileName)
print('#################################')
print('Loading Paths')
print('#################################')
f= open(sFi1eName,'w')
1=0
sline = 'ID|Cost|StartAt|EndAt|Path|Measure'
if nVSet==True: print ('0', sline)
f.write(sline+ '\n')
for sNode0 in G.edges:
    for sNodel in G.edges:
        if sNode0 != sNodel and \
            nx.has ath(G, sNode0, sNode1)==True and \
            nx.shortest ath length(G, \
              source=sNode0, \
              target=sNode1, \
              weight='DistanceMi1es') < nMaxPath:
                l+=1
                sID='{:.0f}'.format(l)
                spath = ','.join(nx.shortest ath(G, \
                  source=sNode0, \
                  target=sNode1, \
                  weight='DistanceMi1es'))
                slength= '{:.6f}'.format(\
                  nx.shortest ath length(G, \
                  source=sNode0, \
                  target=sNode1, \
```

```
                    weight='DistanceMiles'))
                sline = sID + '|"DistanceMiles"|"' + sNode0 + '"|"' \
                + sNodel + '"|"' + spath + '"|' + slength
                if nVSet==True: print (sline)
                f.write(sline + '\n')
                l+=1
                sID='(:.0f}'.format(1)
                spath = ','.join(nx.shortest ath(G, \
                    source=sNode0, \
                    target=sNode1,
                    weight='DistanceMeters'))
                slength= '(:.6f}'.format(\
                    nx.shortest ath_length(G, \
                    source=sNode0, \
                    target=sNode1, \
                    weight='DistanceMeters'))
                sline = sID + '|"DistanceMeters"|"' + sNode0 + '"|"' \
                + sNodel + '"|"' + spath + '"|' + slength
                if nVSet==True: print (sline)
                f.write(sline + '\n')
                l+=1
                sID='{:.0f}'.format(1)
                spath = ','.join(nx.shortest ath(G, \
                    source=sNode0, \
                    target=sNode1, \
                    weight='Cost'))
                slength= '(:.6f}'.format(\
                    nx.shortest ath length(G, \
                    source=sNode0, \
                    target=sNode1, \
                    weight='Cost'))
                sline = sID + '|"Cost"|"' + sNode0 + '"|"' \
                + sNodel + '"|"' + 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:',słD)
sSQL="VACUUM;"
sql.execute(sSQL,conn)
```

```
HQ Rcws      1
Loading : :F:\DS_Practicals_Mqc-it\Practical no 5\practicalnc5J\hillman.db  Table: Asgess_Barehouse
   warehouse_PostCoóe  Warehouse Mame   Barèhouse_Latituóe   Barehouse_Longitude
0                  10          H-  iO              9 .lZ5lé              v2.11731
1                ADh          mn—ABll             97.žZ'8'7Z              —.2.05005
2                 Il          w&-ABl2              5 .10V00              -2.11060
3               PA80          WH-PA80               0.Œ00G0               ù.00000
4                LOO          .Bk-L90              0„00000               0.00000

          Rows

Loadin'g  :  F':\D5_Practicals_Msc'it\Practical  no 5\practicalno5J\hillman.db   Table:  Assess_shop

  Shop_PostCode     Shop_Name Warehouse_PostCode   shmp_Łatitude   Shop_Longitude
0      AB10 1XG   SP-AB10-1XG               AB10       57.144L65        -2.114848
1      ABlG 6     5P''AB10-6RN               QBl0       57.137080        -2'.121487
2      AB'IG 7JB  5P-'AB10-7JB               Aâì0       57.124274        -2.1271Sò
3      ABlÌ SÖN   SP-ABU-5ÕÑ                          57.1.42701        -2.OS32S5
4      ABŁL.6UL   SP-ÄBl1-6UL              .ARlL       57.137547        —2.1'L2-233

Shop Rows :  451
##################################

Storing : F:\DS_Practicals_Msc-it\Practical no                                    g_
##################################
Nodes:     ńŚ 9
Edge'o •   452

                  rac          ⊂                                                    g_
                  ###          ⊓
Loading : F:\bS_#racticaìg_Msc-it\Practical nc 5\practicäldc5J\hillman.db  Taõlъ: 'Asşess_KQ
  KQ_Post€ode  HQ_Mame  NQ_Latitude  HQ_Longitшde
0        KA13  HQ-KA13       55.6553       -4.69921

HQ Rows
Lğaóinq: :  F:\DS_Practiials_Msc-it\Practíca:l  nc  5\practicälno5J\hillman.ób   Table:  Assess_Warehouse
  warehousè_PJstCöde Warehou#e_Ramè  Barвkouse_Lãtitüde'.  Warèhou'se_Lpngitude
0                 IO          H-ABt0              57.13514               -2.1173'1
1               AB11          WH-AB11             57.13875               -2.09089
2               AB12          FH—.AB12            3.?.10100              —2.11Ö.60
3                P 80          WH—P BO              "0. OOOOO              O. 0»0O
4                 80           WH-L80               0. OOOOO              0.GOOO0

.Warehoüse Röös' :   7

Loãdiñg : F:\DS_Pr'ačticals_Msc-it\Pradtical ño 5'\pr'äciicaln'o5J\hiììmañ. ñ  Table: Aèшess_Shop

  shop_Postcodę      Shop_Name 'Barehouse_PostCodo   Shop_Latitude   Bhz@_Łmngitudœ
'0      AB10 1XG.  3P-AB10-'1XG              ABIO       57.144165        -2.Ł14848
ł      'AB1O 6RR   9P-AB10-6RR               AB1O       57.137880       -2.Ł2'1ś87
2'     'AB10 rJB   sE-AB1O-7JB               ABłO       57.124274        -2.Ł271S0
3      ÄØîî NON'   SP-ABl1-5ON               AB11       57.1427Õ1        -2.0SZ2S5
4      Ash  Cuz  Br-Ash-6uz                  Ann        57.í3754'7       -2.112233

9hoB .Rows :  43L


Storing : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5J\Assess_9hipping_Routes.Xml

Nodes: 4SS


9toring : F:\D'5_#racticaìş_Msc-it\Practical nc 5\practicalmc5J\Assoss_9hipping_Routeş:txt

Loading Paths
```

Pratical no 4 K

Aim:-Write a Python program to create Simple forex trading planner from the given data.

Code:-

```
import sqlite3 as sq
import pandas as pd
sInputFileNamel='F:\\DS_Practica1s_Msc-it\\Practica1 no 5\\practica1no5K\\Retrieve-Country-Currency.csv'
sInputFileName2='F:\\DS Practicals Msc-it\\Practical no 5\\practicalno5K\\Retrieve Euro EchangeRates.csv'
sDatabaseName='F:\\DS Practicals Msc-it\\Practica1 no 5\\practica1no5K\\c1ark.db'
conn = sq.connect(sDatabaseName)
sFileName1=sInputFileName1
print('Loading :',sFi1eName1)
CountryRawData d.read csv(sFi1eName1,headem0,low memo False, encoding="latin-1")
CountryRawData.drop duplicates(subset=None, keep='first', inplace=True)
CountryData=CountryRawData
print('Loaded Company :',CountryData.columns.values)
sTab1e='Assess Country'
print('Storing :',sDatabaseName,' Table:',sTable)
CountryData.to_sql(sTable, conn, if_exists="replace")
print(CountryData.head())
print('Rows : ',CountryData.shape[0])
sFi1eName2=sInputFileName2
print('Loading :',sFi1eName2)
ForexRawData=pd.read csv(sFileName2,headem0,low memory=False, encoding="latin-1")
ForexRawData.drop duplicates(subset=None, keep='first', inp1ace=True)
ForexData=ForexRawData.head(5)
prin('Loaded Company :',ForexData.columns.values)
sTab1e='Assess Forex'
print('Storing :',sDatabaseName,' Table:',sTable)
ForexData.to sql(sTable, conn, if exists="replace")
print(ForexData.head())
   print('Rows : ',ForexData.shape[0])
sTab1e='Assess_Forex'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.CodeIn"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess Forex as A;"
odeData=pd.read sql query(sSQL, conn)
for c in range(CodeData.shape[0]):

    sTab1e='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(ForexData)
sTab1e='Assess Forex ' + CodeData['CodeIn'][c]
print('Storing :',sDatabaseName,' Table:',sTable)
ForexData.to sql(sTable, conn, if exists="replace")
print('Rows : ',ForexData.shape[0])

print('Done')
```

Output:-

```
================================ RESTART: F:\DS_Practicals_Msc-it\Practical no 5\practicalno5K\Practicalno5k.py =
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5K\Retrieve-Country-Currency.csv
Loaded Company : ['Country' 'Currency' 'CurrencyCode']
Storing : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5K\clark.db  Table: Assess_Country
                     Country        Currency CurrencyCode
0                Afghanistan  Afghan afghani          AFN
1  Akrotiri and Dhekelia (UK)   European euro          EUR
2     Aland Islands (Finland)   European euro          EUR
3                    Albania    Albanian lek          ALL
4                    Algeria  Algerian dinar          DZD
Rows :  253
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5K\Retrieve_Euro_EchangeRates.csv
Loaded Company : ['Date' 'CodeIn' 'CodeOut' 'Rate']
Storing : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5K\clark.db  Table: Assess_Forex
         Date CodeIn CodeOut      Rate
0  2017-05-05    AUD     AUD  1.000000
1  2017-05-05    AUD     BGN  1.318635
2  2017-05-05    AUD     BRL  2.354167
3  2017-05-05    AUD     CAD  1.017665
4  2017-05-05    AUD     CHF  0.730785
Rows :  5
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5K\clark.db  Table: Assess_Forex
Loading : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5K\clark.db  Table: Assess_Forex & 2x Country > AUD
         Date CodeIn  ...    CurrencyNameOut      Rate
0   2017-05-05    AUD  ...  Australian dollar  1.000000
1   2017-05-05    AUD  ...  Australian dollar  1.000000
2   2017-05-05    AUD  ...  Australian dollar  1.000000
3   2017-05-05    AUD  ...  Australian dollar  1.000000
4   2017-05-05    AUD  ...  Australian dollar  1.000000
..         ...    ...  ...               ...       ...
79  2017-05-05    AUD  ...        Swiss franc  0.730785
80  2017-05-05    AUD  ...        Swiss franc  0.730785
81  2017-05-05    AUD  ...        Swiss franc  0.730785
82  2017-05-05    AUD  ...        Swiss franc  0.730785
83  2017-05-05    AUD  ...        Swiss franc  0.730785

[84 rows x 8 columns]
Storing : F:\DS_Practicals_Msc-it\Practical no 5\practicalno5K\clark.db  Table: Assess_Forex_AUD
Rows :  84
Done
```

Practical no 4 L

Aim;-Write a Python program to process the balance sheet to ensure that only good data is processing

1) Financial

Code;-

```
import sqlite3 as sq
import pandas as pd
sInputFileName='F:\\DS Practicals Msc-it\\Practica1 no 5\\Practica1no5L\\Retrieve Profit And Loss.csv'
sDatabaseName='F:\\DS Practicals Msc-it\\Practica1 no 5\\Practicalno5L\\clark.db'
conn = sq.connect(sDatabaseName)
sFi1eName=sInputFileName
print('Loading :',sFileName)
FinancialRawData=pd.read_csv(sFi1eName,header=0,low_memory=False,encoding="1atin-1")
FinancialData=Financia1RawData
print('Loaded Company :',Financia1Data.columns.values)
sTab1e='Assess-Financials'
print('Storing :',sDatabaseName,' Table:',sTable)
FinancialData.to_sql(sTable, conn, if_exists="replace")
print(FinancialData.head())
print('Rows : ',FinancialData.shape[0])
print('Done')
```

Output:-

```
================================================ RESTART: F:\DS_Practicals_Msc-it\Practical no 5\Pract:
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5L\Retrieve_Profit_And_Loss.csv
Loaded Company : ['QTR' 'TypeOfEntry' 'ProductClass1' 'ProductClass2' 'ProductClass3'
 'Amount' 'QTY']
Storing : F:\DS_Practicals_Msc-it\Practical no 5\Practicalno5L\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
```

2) financiaI_calendar

Code:-

```
import sqlite3 as sq
import pandas as pd
sDatabaseNameIn='F:\\DS Practicals Msc-it\\Practica1 no 5\\Practicalno5L\\c1ark.db'
connln = sq.connect(sDatabaseNameIn)
sDatabaseNameOut='F:\\DS Practicals Msc-it\\Practica1 no 5\\Practica1no5L\\c1ark.db'
connOut = sq.connect(sDatabaseNameOut)
sTab1eIn='Retrieve Date'
sSQL='se1ect * FROM ' + sTableIn + ”,'
sTab1eOut='Assess_Time'
print('Loading :',sDatabaseNameIn,' Tab1e:',sTab1eIn)
```

```
dateRawData=pd.read sql query(sSQL, connIn)
dateData=dateRawData
print('Load Rows : ',dateRawData.shape[0], ' records')
dateData.drop duplicates(subset='FinDate', keep='first', inplace=True)
sTab1eOut='Assess Date'
print('Storing :',sDatabaseNameOut,' Table:',sTableOut)
dateData.to sql(sTableOut, connOut, if exists="replace")
print('Store Rows : ',dateData.shape[0], ' records')
sTab1eIn='Retrieve Time'
sSQL='select * FROM ' + sTableIn + ",'
sTab1eOut='Assess Time'
print('Loading :',sDatabaseNameIn,' Table:',sTab1eIn)
timeRawData d.read_sq1_query(sSQL, connln)
timeData=timeRawData
print('Load Rows : ',timeData.shape[0], ' records')
timeData.drop duplicates(subset=None, keep='first', inp1ace=True)
sTab1eOut='Assess Time'
print('Storing :',sDatabaseNameOut,' Table:',sTableOut)
timeData.to sql(sTab1eOut, connOut, if exists="replace")
print('Store Rows : ',timeData.shape[0], ' records')
print('Done')
```

Output:-

```
========== RESTART: C:\VKHCG\04-Clark\02-Assess\Assess-Calendar.py =======
##################################
Working Base : C:/VKHCG using win32
##################################
################
Loading : C:/VKHCG/04-Clark/01-Retrieve/SQLite/clark.db Table: Retrieve_I
##################################
Load Rows : 50406 records
##################################
################
Storing : C:/VKHCG/04-Clark/01-Retrieve/SQLite/clark.db Table: Assess_Dat
################
##################################
Store Rows : 43830 records
##################################
################
Loading : C:/VKHCG/04-Clark/01-Retrieve/SQLite/clark.db Table: Retrieve_T
##################################
Load Rows : 7196 records
##################################
################
Storing : C:/VKHCG/04-Clark/01-Retrieve/SQLite/clark.db Table: Assess_Tim
################
##################################
Store Rows : 1440 records
##################################
### Done!! ##############################################
```

Practical no 4 M

Aim;-Write a Python program to generate payroll from the given data.

Code:-

```
import sqlite3 as sq
import pandas as pd
Company = '04-Clark'
sInputFileNamel = 'F:\\DS_Practica1s_Msc-it\\Practica1 no 5\\Retrieve-Data_fema1e-names.csv'
sInputFileName2 = 'F:\\DS Practicals Msc-it\\Practica1 no 5\\Retrieve-Data male-names.csv'
sInputFileName3 = 'F:\\DS Practicals Msc-it\\Practical no 5\\Retrieve-Data last-names.csv'
sOutputFileNamel = 'F:\\DS Practicals Msc-it\\Practical no 5\\Assess-Staff.csv'
sOutputFileName2 = 'F:\\DS Practicals Msc-it\\Practica1 no 5\\Assess-Customers.csv'
sDatabaseName ='F:\\DS_Practica1s_Msc-it\\Praetical no 5\\clark.db'
conn = sq.connect(sDatabaseName)
sFileName =sInputFi1eName1
print('Loading :', sFileName)
FemaleRawData = pd.read csv(sFileName, header0, low_memory=False, encoding="latin-1")
FemaleRawData.rename(columns=('NameValues". 'FirstName'}, inplace=True)
Fema1eRawData.drop duplicates(subset=None, keep='first', inplace=True)
FemaleData = FemaleRawData.sample(100)
sTable = 'Assess FemaleName'
print('Storing :', sDatabaseName, ' Table:', sTable)
FemaleData.to sql(sTable, conn, if exists="replace")

print('Rows:', FemaleData.shape[0], 'records')
sFileName =sInputFi1eName2
print('Loading :', sFileName)
MaleRawData = pd.read csv(sFileName, header=0, low memo False, encoding="latin-1")
MaleRawData.rename(columns={'NameValues': 'FirstName'}, inp1ace=True)
Ma1eRawData.drop duplicates(subset=None, keep='first', inplace=True)
MaleData = MaleRawData.sample(100)
sTable = 'Assess MaleName'
print('Storing :', sDatabaseName, ' Table:', sTable)
Ma1eData.to sql(sTable, conn, if exists="replace")
print('Rows:', MaleData.shape[0], 'records')
sFileName =sInputFi1eName3
print('Loading :', sFileName)
SumameRawData = pd.read csv(sFileName, header=0, low memory=False, encoding="latin-1")
SumameRawData.rename(columns=('NameValues". 'LastName'}, inp1ace=True)
SurnameRawData.drop duplicates(subset=None, keep='first', inp1ace=True)
SumameData = SurnameRawData.sample(200)
sTable = 'Assess Surname'
print('Storing :', sDatabaseName, ' Table:', sTable)
SumameData.to sql(sTable, conn, if exists="replace”)
print('Rows:', SurnameData.shape[0], 'records')
sTable = 'Assess FemaleName & Assess MaleName'
print('Loading :', sDatabaseName, ' Table:', sTable)
sSQL = ”""
    SELECT DISTINCT
        A.FirstName,
```

```
        'Female' AS Gender
    FROM
        Assess_FemaleName AS A
    UNION
    SELECT DISTINCT
        A.FirstName,
        'Male' AS Gender
    FROM
        Assess_MaleName AS A;

FirstNameData = pd.read_sql_query(sSQL, conn)


sTable = 'Assess_FirstName'
print('Storing :', sDatabaseName, ' Table:', sTable)
FirstNameData.to_sql(sTable, conn, if_exists="replace")



sTable = 'Assess_FirstName x2 & Assess_Surname'
print('Loading :', sDatabaseName, ' Table:', sTable)
sSQL = """
    SELECT DISTINCT
        A.FirstName,
        B.FirstName AS SecondName,
        C.LastName,
        A.Gender
    FROM
        Assess_FirstName AS A,
        Assess_FirstName AS B,
        Assess_Surname AS C
    WHERE
        A.Gender = B.Gender
    AND
        A.FirstName <> B.FirstName;

PeopleRawData = pd.read_sql_query(sSQL, conn)
PeoplelData = PeopleRawData.sample(10000)


sTable = 'Assess_FirstName & Assess_Surname'
print('Loading :', sDatabaseName, ' Table:', sTable)
sSQL = """
    SELECT DISTINCT
        A.FirstName,
        '' AS SecondName,
        B.LastName,
        A.Gender
    FROM
        Assess_FirstName AS A,
```

```
            Assess Surname AS B;
"""PeopleRawData = pd.read sql query(sSQL, conn)
People2Data = PeopleRawData.sample(10000)
PeopleData = pd.concat([PeoplelData, People2Data], ignore index=True)
print(Peop1eData)
   sTable = 'Assess People'
print('Storing :', sDatabaseName, ' Table:', sTable)
PeopleData.to_sql(sTable, conn, if_exists="replace")
   sOutputFileName = "Practical no 5\\" +sTab1e + '.csv'
sFileName = sOutputFileName
print('Storing :', sFileName)
Peop1eData.to csv(sFi1eName, index=Fa1se)
print('Done')
```

Output:-

```
================================================ RESTART: F:\DS_Practicals_Msc-it\Practical no 5\Prac
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Retrieve-Data_female-names.csv
Storing : F:\DS_Practicals_Msc-it\Practical no 5\clark.db  Table: Assess_FemaleName
Rows: 100 records
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Retrieve-Data_male-names.csv
Storing : F:\DS_Practicals_Msc-it\Practical no 5\clark.db  Table: Assess_MaleName
Rows: 100 records
Loading : F:\DS_Practicals_Msc-it\Practical no 5\Retrieve-Data_last-names.csv
Storing : F:\DS_Practicals_Msc-it\Practical no 5\clark.db  Table: Assess_Surname
Rows: 200 records
Loading : F:\DS_Practicals_Msc-it\Practical no 5\clark.db  Table: Assess_FemaleName & Assess_MaleName
Storing : F:\DS_Practicals_Msc-it\Practical no 5\clark.db  Table: Assess_FirstName
Loading : F:\DS_Practicals_Msc-it\Practical no 5\clark.db  Table: Assess_FirstName x2 & Assess_Surname
Loading : F:\DS_Practicals_Msc-it\Practical no 5\clark.db  Table: Assess_FirstName & Assess_Surname
        FirstName SecondName  LastName  Gender
0      Florentina     Sherie   Eastman  Female
1           Angel         Fe   Belcher  Female
2           Janel     Jesusa     Bryan  Female
3           Angel    Chelsey    Thorne  Female
4        Benedict      Merle  Aldridge    Male
...           ...        ...       ...     ...
19995     Sheldon                 Toro    Male
19996      Jesusa               Wallis  Female
19997       Twyla                Gates  Female
19998       Tresa               Cooley  Female
19999        Sook               Thorne  Female

[20000 rows x 4 columns]
Storing : F:\DS_Practicals_Msc-it\Practical no 5\clark.db  Table: Assess_People
Storing : Practical no 5\Assess_People.csv
```

	A	B	C	D
1	FirstName	SecondNan	LastName	Gender
2	Casey	Harland	Finnegan	Male
3	Alleen	Shirley	Almond	Female
4	Diego	Hershel	Etheridge	Male
5	Lesha	Iona	Otis	Female
6	Alexander	Cruz	Pike	Male
7	Gary	Britta	Negrete	Female
8	Nettie	Hye	Pauley	Female
9	Kerry	Alexander	Cash	Male
10	Arielle	Robbie	Gorman	Female
11	Nestor	Casey	Fogle	Male
12	Concepcior	Melodie	Harry	Female
13	Tamesha	Robbie	Sears	Female
14	See	Shirley	Mcinnis	Female

Practival no 5
Aim:-Build the time hub, links and satellites.

Code:-

```
import os
from datetime import datetime, timedelta
import pandas as pd
import sqlite3 as sq
import uuid
from pytz import timezone, all_timezones
Base 'F://MCSIT add documentsDS Practicals Msc-it//Practica1 no 6//Bui1d time hub links and_sate1lites'
Company = '01-Vermeu1en'
sDataBaseDir = os.path.join(Base, Company, '03-Process', 'SQLite')
sDataVaultDir = os.path.join(Base, '88-DV')
os.makedirs(sDataBaseDir, exist_ok=True)
os.makedirs(sDataVaultDir, exist_ok=True)
connl = sq.connect(os.path.join(sDataBaseDir, 'Hi11man.db'))
conn2 = sq.connect(os.path.join(sDataVaultDir, 'datavault.db'))
base = datetime(2018, 1, 1, 0, 0, 0)
numUnits = 7 * 24
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)]
batch_size = 100
time_data = []
for t, i in enumerate(date_list, start=1):
    now_utc = i.replace(tzinfo=timezone('UTC'))
    sDateTime = now_utc.strftime("%Y-%m-%d %H:%M:%S")
    sDateTimeKey = sDateTime.replace(' ', '-').replace(".', '-')
    IDNumber = str(uuid.uuid4())

    TimeLine = (
        'ZoneBaseKey': 'UTC',
        'IDNumber'. IDNumber,
        'nDateTimeValue': now_utc,
        'DateTimeValue': sDateTime,
        'DateTimeKey': sDateTimeKey

    time_data.append(TimeLine)
    print(f'Processed DateTime: (sDateTime} (ID: {IDNumber})")
    if t % batch_size == 0 or t == 1en(date_1ist):
        TimeFrame = pd.DataFrame(time_data)
        TimeFrame.set_index(['IDNumber'], inp1ace=True)
        TimeFrame.to_sql('Process-Time', connl, if_exists="append", index=True)
        time_data.clear()

if'IDNumber' in TimeFrame.columns:
    TimeHub = TimeFrame[['IDNumber', 'ZoneBaseKey', 'DateTimeKey', 'DateTimeValue']]
    TimeHubIndex = TimeHub.set_index(['IDNumber'], inp1ace=False)
    TimeHubIndex.to_sql('Hub-Time', conn2, if_exists="replace", index=True)
else:
    print("Error: 'IDNumber' not found in the columns of TimFrame")
```

```
for zone in all timezones:
    time zone data = []
    for t, row in TimeFrame.iterrows():
        now date = row['nDateTimeVa1ue']
        DateTimeKey = row['DateTimeKey']
        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")

        IDZoneNumber = str(uuid.uuid4())
        TimeZoneLine = (
            'ZoneBaseKey". 'UTC',
            'IDZoneNumber': IDZoneNumber,
            'DateTimeKey". DateTimeKey,
            'UTCDateTimeValue“. sDateTime,
            'Zone': zone,
            'DateTimeValue". sZoneDateTime

        time_zone_data.append(TimeZoneLine)
    TimeZoneFrame = pd.DataFrame(time zone data)
    TimeZoneFrame.set_index(['IDZoneNumber'], inp1ace=True)
    TimeZoneFrame.to sql(fProcess-Time-(zone.replace(”/", "-").replace(" ", "")}', connl,
if exists="replace", index=True)
    TimeZoneFrame.to sql(fSatellite-Time-(zone.replace("/", "-").replace(" ", "")}', conn2,
if exists="replace", index=True)
sq.execute("VACUUM;", connl)
sq.execute(”VACUUM;”, conn2)

print('Done')
```

Output:-

```
========================= RESTART: F:\DS_Practicals_Msc-it\Practical no 6\Build_time_hub_links_and_s
Processed DateTime: 2018-01-01 00:00:00 (ID: 21ba70bc-f951-46cb-beda-2fff02344402)
Processed DateTime: 2017-12-31 23:00:00 (ID: b869527e-e0ba-4027-9dbb-05dfeb655bcb)
Processed DateTime: 2017-12-31 22:00:00 (ID: 540d856c-b2cd-4d26-b6b5-7d203a5254ea)
Processed DateTime: 2017-12-31 21:00:00 (ID: dec4fd50-118c-4f56-be5e-de196e0c057d)
Processed DateTime: 2017-12-31 20:00:00 (ID: 80a16798-19c2-4b3a-a0f9-18463581fb46)
Processed DateTime: 2017-12-31 19:00:00 (ID: 7c183e5c-48b2-4d22-b83b-7695fcc27b08)
Processed DateTime: 2017-12-31 18:00:00 (ID: 5d0976c5-21b9-48f3-a80d-482fa9fd97b8)
Processed DateTime: 2017-12-31 17:00:00 (ID: fd732a34-55e3-4f22-90a0-4516d0943a18)
Processed DateTime: 2017-12-31 16:00:00 (ID: 21858a51-768e-4236-be91-f433569e1513)
Processed DateTime: 2017-12-31 15:00:00 (ID: 5b40554f-6030-4a16-8509-9cc8e725aa3b)
Processed DateTime: 2017-12-31 14:00:00 (ID: e213e170-53b5-4d5f-8bdd-14d9f33ee9c3)
Processed DateTime: 2017-12-31 13:00:00 (ID: 19392b01-71ff-4a90-af76-c161d518ace4)
Processed DateTime: 2017-12-31 12:00:00 (ID: 4f723398-2c7c-4c26-8246-bb32dbf81c56)
Processed DateTime: 2017-12-31 11:00:00 (ID: a5032ce8-6817-4848-b99e-4a0fafc74250)
Processed DateTime: 2017-12-31 10:00:00 (ID: f16f0b0e-a719-4a1c-adfa-e618de9e8335)
Processed DateTime: 2017-12-31 09:00:00 (ID: 8c97e59a-0939-4dc1-8c34-1348e32a81fa)
Processed DateTime: 2017-12-31 08:00:00 (ID: 755c0662-c261-4a67-b66a-0fd5d9b54114)
Processed DateTime: 2017-12-31 07:00:00 (ID: 278328a4-56ef-4324-95da-6a42678f81ef)
Processed DateTime: 2017-12-31 06:00:00 (ID: 829d2d0a-61f4-465c-b9d7-3cc80cea4efe)
Processed DateTime: 2017-12-31 05:00:00 (ID: 77c84e7b-57ce-40d8-9bff-967acb5dd737)
Processed DateTime: 2017-12-31 04:00:00 (ID: 2d66343a-2e36-4e42-acc1-c56f161b93b9)
```

1) Vehicle

Code:-

```
import os
import pandas as pd
import sqlite3 as sq
import uuid
from pandas.io import sql
pd.options.mode.chained assignment = None
Base="F:\\DS Practicals Msc-it\\Practica1 no 6"
sDataBaseDir = os.path.join(Base,'vehicle', '03-Process/SQLite')
os.makedirs(sDataBaseDir, exist ok=True)
sDatabaseName = os.path.join(sDataBaseDir, 'Hi11man.db')
connl = sq.connect(sDatabaseName)
sDataVaultDir = os.path.join(Base,'vehicle', '88-DV')
os.makedirs(sDataVaultDir, exist ok=True)
sDatabaseName = os.path.join(sDataVaultDir, 'datavault.db')
conn2 = sq.connect(sDatabaseName)
sFileName = "F:\\DS Practicals Msc-it\\Practica1 no 6\\Vehic1eData.csv"
print('Loading:', sFileName)
VehicleRaw = pd.read csv(sFileName, header0, low memo      False, encoding="1atin-1")


sTable = 'Process_Vehic1es'
print('Storing:', sDatabaseName, 'Table:', sTable)
Vehic1eRaw.to sql(sTable, connl, if exists="replace")


VehicleRawKey = Vehic1eRaw[['Make', 'Model']].copy()
VehicleKey = Vehic1eRawKey.drop duplicates()
Vehic1eKey['ObjectKey'] = VehicleKey.apply(lambda row: f'({row['Make'].strip().replace(' ', '-
').replace('/', '-').lower()})-((row['Model'].strip().replace(' ', '-').replace('/', '-').lower()})', axis=1)
Vehic1eKey['ObjectType'] = 'vehicle'
Vehic1eKey['ObjectUUID'] = VehicleKey.apply(lambda row: str(uuid.uuid4()), axis=1)


VehicleHub = Vehic1eKey[['ObjectType', 'ObjectKey', 'ObjectUUID']].copy()
VehicleHub.index.name = 'ObjectHubID'
sTable = 'Hub-Object-Vehicle'
print('Storing:', sDatabaseName, 'Table:', sTable)
Vehic1eHub.to sql(sTab1e, conn2, if exists="replace")


VehicleSatellite = VehicleKey[['ObjectType', 'ObjectKey', 'ObjectUUID', 'Make',
'Model']].copy()
VehicleSatellite.index.name = 'ObjectSatelliteID'
sTable = 'Satellite-Object-Make-Model'
print('Storing:', sDatabaseName, 'Table:', sTable)
Vehic1eSate11ite.to sq1(sTab1e, conn2, if exists="replace")
sView = 'Dim-Object'
print('Storing:', sDatabaseName, 'View:', sView)
```

```
sSQL = fi"""
    CREATE VIEW IF NOT EXISTS [(sView}] AS
    SELECT DISTINCT
        H.ObjectType,
        H.ObjectKey AS VehicleKey,
        TRIM(S.Make) AS VehicleMake,
        TRIM(S.Model) AS VehicleModel
    FROM
        [Hub-Object-Vehicle] AS H

        [Satellite-Object-Make-Model] AS S
    ON
        H.ObjectType = S.ObjectType
    AND
        H.ObjectUUID = S.ObjectUUID;

sql.execute(sSQL, conn2)
print('Loading:', sDatabaseName, 'Table:', sView)
sSQL = fi""'
    SELECT DISTINCT
        VehicleMake,
        VehicleModel
    FROM
        [(sView}]
    ORDER BY
        VehicleMake, VehicleModel;

DimObjectData = pd.read  sq1_query(sSQL, conn2)
DimObjectData.index.name = 'ObjectDimID'
DimObjectData.sort   values(['Vehic1eMake', 'VehicleModel'], inp1ace=True,  ascending=True)

print(DimObjectData)
print('Vacuum Databases')
sq1.execute("VACUUM;", connl)
sq1.execute("VACUUM;", conn2)
conn1.close()
conn2.close()
print('Done')
```

Output:-

```
Loading: F:\DS_Practicals_Msc-it\Practical no 6\vehicle\88-DV\datavault.db Table: Dim-Object
                VehicleMake                      VehicleModel
ObjectDimID
0                AM General                DJ Po Vehicle 2WD
1                AM General                  FJ8c Post Office
2                AM General              Post Office DJ5 2WD
3                AM General              Post Office DJ8 2WD
4          ASC Incorporated                              GNX
...                     ...                              ...
3880                  smart              fortwo convertible
3881                  smart                    fortwo coupe
3882                  smart   fortwo electric drive cabriolet
3883                  smart  fortwo electric drive convertible
3884                  smart       fortwo electric drive coupe

[3885 rows x 2 columns]
Vacuum Databases
```

3) HumanEnvironmentInteraction

Code:-

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import uuid
connl = sq.connect('Practical no 6\\Vermeulen.db')
conn2 = sq.connect('Practical no 6\\datavault.db')
t = 0
tMax = 360 * 180
LocationFrame = pd.DataFrame()
for Longitude in range(-180, 180, 10):
    for Latitude in range(-90, 90, 10):
        t += 1
        IDNumber = str(uuid.uuid4())
        LocationName = fL(round(Longitude, 3) * 1000:07d}-(round(Latitude, 3) *
1000:07d}'
        print(fCreate: (t} of (tMax}: (LocationName}')

        LocationLine = (
            'ObjectBaseKey': ['GPS'],
            'IDNumber'. [IDNumber],
            'LocationNumber': [str(t)],
            'LocationName'. [LocationName],
            'Longitude'. [Longitude],
            'Latitude': [Latitude]          }
        LocationRow = pd.DataFrame(LocationLine)
        LocationFrame = pd.concat([LocationFrame, LocationRow], ignore index=True)
LocationHubIndex = LocationFrame.set index(['IDNumber'], inplace=False)
sTable = 'Process-Location'
print(fStoring: Datavau1t.db Table: (sTable}')
LocationHubIndex.to sql(sTable, connl, if_exists="replace")
sTable = 'Hub-Location'
print(fStoring: Datavau1t.db Table: (sTable}')
LocationHubIndex.to sql(sTable, conn2, if_exists="replace")
print('Vacuum Databases')
connl.execute("VACUUM;")
conn2.execute("VACUUM;")
print('Done')
```

```
Create: 643 of 64800: L0170000-0030000
Create: 644 of 64800: L0170000-0040000
Create: 645 of 64800: L0170000-0050000
Create: 646 of 64800: L0170000-0060000
Create: 647 of 64800: L0170000-0070000
Create: 648 of 64800: L0170000-0080000
Storing: Datavault.db Table: Process-Location
Storing: Datavault.db Table: Hub-Location
Vacuum Databases
Done
```

4) Forecasting

Code:-

```
import sys
import os
import sqlite3 as sq
import quandl
import pandas as pd
# Base directory setup
BASE_DIR = 'PRACTICALS'
COMPANY = 'Practical no 6'
print('##################################')
print('Working Base:', BASE_DIR, 'using', sys.platform)
print('##################################')
# File paths and directories
INPUT_FILE = 'VKHCG_Shares.csv'
OUTPUT_FILE = 'Shares.csv'
# Directory paths for storing processed data
PROCESS_DIR = I'(BASE_DIR}/(COMPANY}/03-Process/SQLite"
RETRIEVE_DIR = I"(BASE_DIR}/(COMPANY}/01-Retrieve/01-EDS/02-Python"
ASSESS_DIR = fi'(BASE_DIR}/(COMPANY}/02-Assess/01-EDS/02-Python"
PROCESS_PYTHON_DIR = I'{BASE_DIR}/{COMPANY}/03-Process/01-EDS/02-Python"
# Ensure directories exist
os.makedirs(PROCESS_DIR, exist_ok=True)
os.makedirs(RETRIEVE_DIR, exist_ok=True)
os.makedirs(ASSESS_DIR, exist_ok=True)
os.makedirs(PROCESS_PYTHON_DIR, exist_ok=True)
# Database connection
DB_NAME = I"(PROCESS_DIR}/c1ark.db"
conn = sq.connect(DB_NAME)
# Import Share Names Data
input_file_ath = fi'F:\\DS_Practicals_Msc-it\\Practica1 no 6\\VKHCG_Shares.csv"
print('##################################')
print('Loading:', input_file_ath)
print('##################################')
# Read CSV data
raw_data = pd.read_csv(input_file_ath, header0, low_memo_False, encoding="latin-1")
raw_data.drop_duplicates(inplace=True)
print('Rows :', raw_data.shape[0])
print('Columns:', raw_data.shape[1])
print('################')
# Save raw data to different directories
def save_data(data, dir_ath, prefix):
    output_file_ath = f'(dir_ath}/(prefix} (OUTPUT_FILE}"
    print('##################################')
    print('Storing:', output_file_ath)
    print('##################################')
    data.to_csv(output_file_ath, index=False)
    print('##################################')
save_data(raw_data, RETRIEVE_DIR, 'Retrieve_')
save_data(raw_data, ASSESS_DIR, 'Assess_')
```

```
save data(raw data, PROCESS PYTHON DIR, 'Process_')
# Import Shares Data Details
n shares = raw data.shape[0]
# Fetch and process share data from Quandl
for s share in range(n shares):
    s_share_name = str(raw_data['Shares'][s_share])
    share data = quandl.get(s_share name)

    # Add 'UnitsOwn' and 'ShareCode' columns to share data
    units own = raw data['Units'][s share]
    share data['UnitsOwn'] = units own
    share data['ShareCode'] = s share name
    print('#################')
    print('Share   :', s share name)
    print('Rows    :', share_data.shape[0])
    print('Columns:', share data.shape[1])
    print('#################')
    # Store share data in SQLite database
    s table = str(raw data['sTable'][s share])
    print('Storing:', DB_NAME, 'Table:', s_tab1e)
    share data.to sql(s table, conn, if exists="replace")
    # Save share data to CSV in different directories
    save data(share data, RETRIEVE DIR, fRetrieve (s table.replace("/", "-")} ')
    save data(share data, ASSESS DIR, fAssess (s table.replace("/", "-")} ')
    save data(share data, PROCESS PYTHON DIR, fProcess (s table.replace("/", "-")} ')
print('### Done!! ##############################################')
```

```
-----------------------------
Working Base: PRACTICALS using win32
################################
################################
Loading: F:\DS_Practicals_Msc-it\Practical no 6\VKHCG_Shares.csv
################################
Rows    : 10
Columns: 3
################
################################
Storing: PRACTICALS/Practical no 6/01-Retrieve/01-EDS/02-Python/Retrieve_Shares.csv
################################
################################
################################
Storing: PRACTICALS/Practical no 6/02-Assess/01-EDS/02-Python/Assess_Shares.csv
################################
################################
################################
Storing: PRACTICALS/Practical no 6/03-Process/01-EDS/02-Python/Process_Shares.csv
-----------------------------
```

Practical no 6
Aim:-Transforming Data

1) Transform Superstep_A

Code:-

```
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained assignment = None
connl = sq.connect("F:\\DS Practicals Msc-it\\Practica1 no 7\\Vermeu1en.db")
conn2 = sq.connect("F:\\DS_Practica1s_Msc-it\\Practical no 7\\datavault.db")
conn3 = sq.connect("F:\\DS Practicals Msc-it\\Practical no 7\\datawarehouse.db")
print('Time Category')
print('UTC Time')
birth date utc = datetime(1960, 12, 20, 10, 15, 0).replace(tzinfo=timezone('UTC'))
birth date utc str = birth date utc.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
print(birth date_utc str)

print('Birth Date in Reykjavik:')
birth_zone = 'Atlantic/Reykjavik'
birth date local = birth date_utc.astimezone(timezone(birth zone))
birth date 1oca1_str = birth date local.strftime("°%Y-%m-%d %H:%M:%S (%Z) (%z)")
print(birth date_1oca1 str)

time_zone_id = str(uuid.uuid4())
date time key = birth date utc.strftime("%Y-%m-%d-%H-%M-%S")
time data = {
    'ZoneBaseKey': ['UTC'],
    'IDNumber': [time zone id],
    'DateTimeKey'. [date time key],
    'UTCDateTimeValue': [birth date utc],
    'Zone'. [birth_zone],
    'DateTimeValue“. [birth date local str]

time frame = pd.DataFrame.from dict(time data)

time hub time_frame[['IDNumber', 'ZoneBaseKey', 'DateTimeKey',
'DateTimeValue']].set index('IDNumber')
time_hub.to_sq1('Hub-Time-Gunnarsson', conn2, if_exists='rep1ace')
time hub.to sql('Dim-Time-Gunnarsson', conn3, if exists='rep1ace')

time satellite time frame[['IDNumber', 'DateTimeKey', 'Zone',
'DateTimeValue']].set index('IDNumber')
zone fix = birth_zone.replace(' ', '-').replace('/', '-')
satellite table name = fSate11ite-Time-(zone fix}-Gunnarsson'
time_sate11ite.to_sql(sate11ite_tab1e_name, conn2, if_exists='rep1ace')
time satellite.to_sq1(fDim-Time-(zone fix}-Gunnarsson', conn3, if exists='replace')
```

```
print()
print('Person Category')
first name = 'Gu6mundur'
last name = 'Gunnarsson'
print(fName: {first_name} (1ast_name}')
print(fBirth Date: (birth date_1oca1.strftime("%Y-%m-%d %H:%M:%S”)}')
print(fBirth Zone: {birth_zone}')
print(fUTC Birth Date: (birth date utc.strftime("%Y-%m-%d %H:%M:%S”)}')

person id = str(uuid.uuid4())
person data = {
    'IDNumber': [person_id],
    'FirstName". [first name],
    'LastName': [1ast_name],
    'Zone“. ['UTC'],
    'DateTimeValue". [birth date utc str]

person frame = pd.DataFrame.from dict(person data)
person_hub = person_frame.set_index('IDNumber')
person hub.to sql('Hub-Person', conn2, if exists='replace')
person_hub.to_sql('Dim-Person', conn3, if_exists='rep1ace')
print('All Data Stored Successfully!')
```

Op:-

```
>
  ==================================================
  Time Category
  UTC Time
  1960-12-20 10:15:00 (UTC) (+0000)
  Birth Date in Reykjavik:
  1960-12-20 10:15:00 (GMT) (+0000)

  Person Category
  Name: Guðmundur Gunnarsson
  Birth Date: 1960-12-20 10:15:00
  Birth Zone: Atlantic/Reykjavik
  UTC Birth Date: 1960-12-20 10:15:00
  All Data Stored Successfully!
>
```

2) Transform Superstep_B

Code:-

```
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained assignment = None
connl = sq.connect("F:\\DS Practicals Msc-it\\Practical no 7\\Vermeulen.db")
conn2 = sq.connect("F:\\DS Practicals Msc-it\\Practica1 no 7\\datawarehouse.db")
print(")
print('Time Dimension')
BIRTH_ZONE = 'Atlantic/Reykjavik'
BIRTH DATE UTC = datetime(1960, 12, 20, 10, 15, 0).replace(tzinfomimezone('UTC'))
BIRTH_DATE_UTC_STR = BIRTH_DATE_UTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
 birth date local = BIRTH DATE UTC.astimezone(timezone(BIRTH ZONE))
birth date local str = birth date local.strftime("°%Y-%m-%d %H:%M:%S (%Z) (%z)")
time id = str(uuid.uuid4())
time_data = (
      'TimeID“. [time id],
        'UTCDate': [BIRTH_DATE_UTC.strftime("%Y-%m-%d %H:%M:%S")],
      'LocalTime”. [birth date local.strftime("%Y-%m-%d %H:%M:%S")],
      'TimeZone': [BIRTH ZONE]

time frame = pd.DataFrame(time data)
time_frame.set_index('TimeID').to_sq1('Dim-Time', connl, if_exists='rep1ace')
time frame.set index('TimeID').to sql('Dim-Time', conn2, if exists='rep1ace')
print(time_frame)
print(")
print('Dimension Person')
first name = 'Gu6mundur'
last name = 'Gunnarsson'
person_id = str(uuid.uuid4())


person data = (
      'PersonID': [person id],
      'FirstName'. [first name],
      'LastName': [last name],
      'Zone'. ['UTC'],
      'DateTimeValue'. [BIRTH DATE UTC STR]

person frame = pd.DataFrame(person data)
person_frame.set_index('PersonID').to_sq1('Dim-Person', connl, if_exists='replace')
person frame.set index('PersonID').to sql('Dim-Person', conn2, if exists='rep1ace')

print(person_frame)
```

```
print('')
print('Fact - Person - Time')

fact_id = str(uuid.uuid4())
fact_data = (
    'IDNumber': [fact_id],
    'IDPersonNumber“. [person_id],
    'IDTimeNumber': [time_id]

fact_frame = pd.DataFrame(fact_data)


fact_frame.set_index('IDNumber').to_sql('Fact-Person-Time', connl, if_exists='rep1ace')
fact_frame.set_index('IDNumber').to_sql('Fact-Person-Time', conn2, if_exists='replace')

print(fact_frame)

print('All Data Stored Successfully!')
```

Output:-

```
Time Dimension
                                TimeID  ...            TimeZone
0  75f98227-d9b1-48f5-8052-f35b47f5968a  ...  Atlantic/Reykjavik

[1 rows x 4 columns]

Dimension Person
                              PersonID  ...                      DateTimeValue
0  8958525d-da17-4f6e-81a4-77789db2c5c3  ...  1960-12-20 10:15:00 (UTC) (+0000)

[1 rows x 5 columns]

Fact - Person - Time
                              IDNumber  ...                          IDTimeNumber
0  73de456b-5b9a-4cd1-8794-8f33c93ef0cc  ...  75f98227-d9b1-48f5-8052-f35b47f5968a

[1 rows x 3 columns]
All Data Stored Successfully!
```

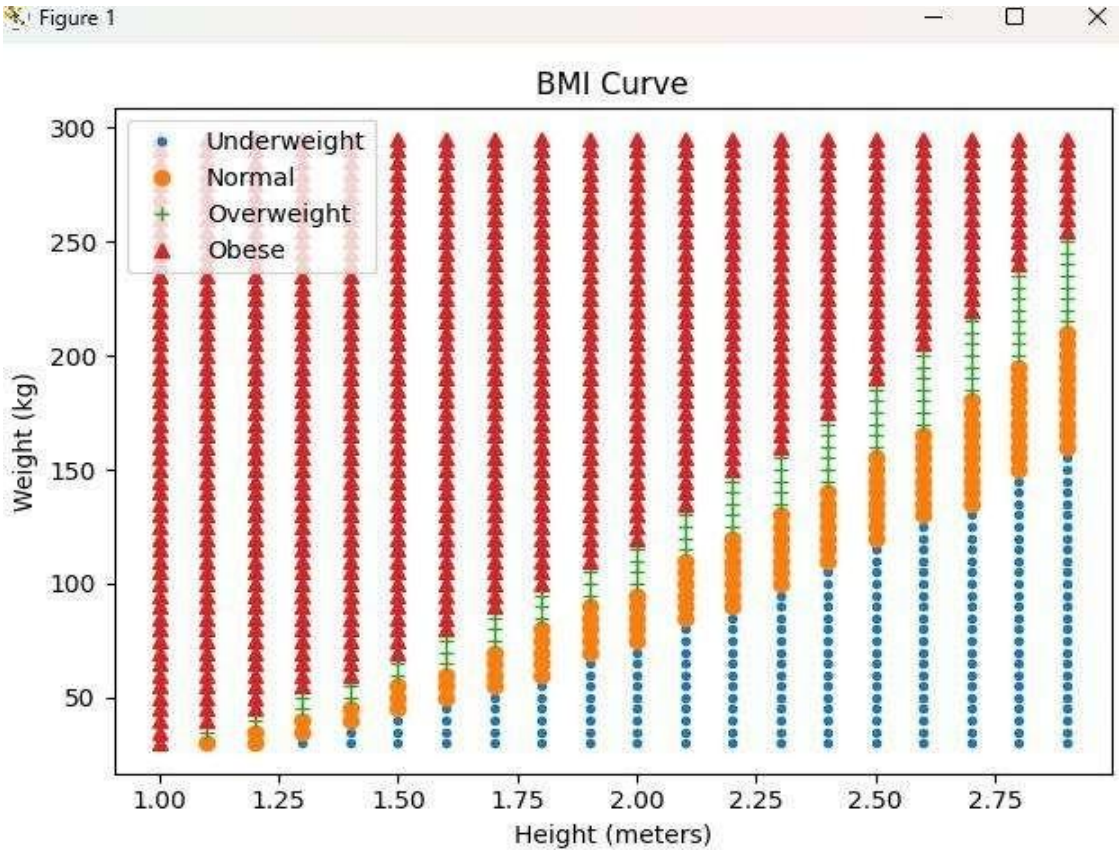
3) SimpleRegression

Code:-

```
import pandas as pd
import sqlite3 as sq
import matp1ot1ib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean squared error, r2 score
connl = sq.connect("F:\\DS Practicals Msc-it\\Practical no 7\\Vermeulen.db")
conn2 = sq.connect("F:\\DS Practicals Msc-it\\Practical no 7\\datavau1t.db")
conn3 = sq.connect("F:\\DS Practicals Msc-it\\Practica1 no 7\\datawarehouse.db")
t = 0
bmi data = []

for height select in range(100, 300, 10):
    for weight_select in range(30, 300, 5):
        height = round(height select / 100, 3)
        weight = int(weight select)
        bmi = weight / (height * height)

        ifbmi <= 18.5:
            bmi result = 1
        elif 18.5 < bmi < 25:
            bmi result = 2
        elif 25 <= bmi < 30:
            bmi result = 3
        else:
            bmi_result = 4


        bmi data.append((
            'PersonID': t,
            'Height': height,
            'Weight". weight,
            'BMI': bmi,
            'Indicator“. bmi result

        t += 1
bmi df= pd.DataFrame(bmi data)
bmi df.set index('PersonID', inp1ace=True)
bmi df.to sql('Transform-BMI', connl, if exists='rep1ace')
bmi_df.to_sq1('Person-Satellite-BMI', conn2, if_exists='rep1ace')
bmi df.to sql('Dim-BMI', conn3, if exists='replace')
print('BMI Data Saved Successfully!')
fig = plt.figure()
for indicator, marker, label in zip([1, 2, 3, 4], [".", "o", "+", "^"],
                                    [”Underweight", "Normal", "Overweight",
"Obese”]): subset = bmi df[bmi df['Indicator'] indicator]
    plt.plot(subset['Height'], subset['Weight'], marker, 1abe1=1abe1)
```

```
plt.title("BMI Curve")
plt.xlabel("Height (meters)")
plt.ylabel("Weight (kg)")
plt.legend()
p1t.tight layout()
plt.show()
diabetes = datasets.load diabetes()
diabetes_X = diabetes.data[:, np.newaxis, 2]
diabetes X train = diabetes X[:-30]
diabetes X test = diabetes X[-30:]
diabetes_y train = diabetes.target[:-30]
diabetes_y test = diabetes.target[-30:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X train, diabetes_y train)
diabetes red = regr.predict(diabetes_X_test)
print('Coefficients:', regr.coef
print("Mean squared error: %.2f" % mean squared error(diabetes_y test, diabetes y red))
print('Variance score: %.2f % r2 score(diabetes_y test, diabetes y red))
plt.scatter(diabetes X test, diabetes_y test, co1om'black', 1abe1="Actua1 Data")
plt.plot(diabetes_X_test, diabetes red, co1or='b1ue', linewidth=2, 1abel="Predicted Data")
plt.title("Diabetes - BMI vs Disease Progression")
p1t.xlabel("BMI")
plt.ylabel("Disease Progression")
p1t.legend()
p1t.tight layout()
p1t.show()
```

Output:-



4)

4) Building_a_DataWarehouse

Code:-

```
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained assignment = None
connl = sq.connect("F:\\DS Practicals Msc-it\\Practical no 7\\Vermeulen.db")
conn2 = sq.connect("F:\\DS Practicals Msc-it\\Practical no 7\\datavault.db")
conn3 = sq.connect("F:\\DS_Practica1s_Msc-it\\Practical no 7\\datawarehouse.db")
date query = ”SELECT DateTimeValue FROM [Hub-Time];"
date_data_raw = pd.read_sql_query(date_query, conn2)
date data = date data raw.head(1000)

print(date data)
print('\n Time Dimension \n ')

time frames = []
birth_zones = ['Atlantic/Reykjavik', 'Europe/London', 'UTC']

for i, row in date data.iterrows():
    birth date utc datetime.strptime(row['DateTimeValue'], %Y-%m-
%d %H:%M:%S").replace(tzinfo=timezone('UTC'))
    for zone in birth_zones:
        birth date local = birth date utc.astimezone(timezone(zone))
        time_id = str(uuid.uuid4())
        time entry = (
            'TimeID': time id,
            'UTCDate". birth_date utc.strftime("%Y-%m-%d %H:%M:%S"),
            'LocalTime': birth date local.strftime(”%Y-%m-%d %H:%M:%S"),
            'TimeZone': zone

        time_frames.append(time_entry)
time frame = pd.DataFrame(time frames)
time frame.set index('TimeID').to sql('Dim-Time', connl, if exists='replace')
time frame.set index('TimeID').to sql('Dim-Time', conn3, if exists='rep1ace')
person query = ”"""
SELECT
    FirstName,
    LastName
FROM [Hub-Person];

person data raw = pd.read sq1_query(person query, conn2)
person data = person data raw.head(1000)

print('\n Dimension Person \n ')
```

```
person_frames = []

for i, row in person_data.iterrows():

    first_name = row['FirstName']
    second_name = row.get('SecondName', '') or ''
    Iast_name = row['LastName']
    person_id = str(uuid.uuid4())
    person_entry = {
        'PersonID'. person_id,
        'FirstName*. first_name,
        'LastName'. Iast_name,
        'Zone'. 'UTC',

    person_frames.append(person_entry)

 person_frame = pd.DataFrame(person_frames)

person_frame.set_index('PersonID').to_sql('Dim-Person', connl, if_exists='replace')
person_frame.set_index('PersonID').to_sql('Dim-Person', conn3, if_exists='replace')

print('All Data Stored Successfully!')
```

Code:-

```
==================================
           DateTimeValue
0    2018-01-01 00:00:00
1    2017-12-31 23:00:00
2    2017-12-31 22:00:00
3    2017-12-31 21:00:00
4    2017-12-31 20:00:00
..                   ...
995  2017-11-20 13:00:00
996  2017-11-20 12:00:00
997  2017-11-20 11:00:00
998  2017-11-20 10:00:00
999  2017-11-20 09:00:00

[1000 rows x 1 columns]

 Time Dimension


 Dimension Person

All Data Stored Successfully!
```

Practical no 7
Aim:-Organizing Data

1) Horizontal Style

Code:-

```
import pandas as pd
import sqlite3 as sq
datawarehouse_db = "F:\\DS_Practica1s_Msc-it\\Practical no 8\\datawarehouse.db"
datamart db = "F:\\DS Practicals Msc-it\\Practical no 8\\datamart.db"
connl = sq.connect(datawarehouse db)
conn2 = sq.connect(datamart db)
sTable = "Dim-BMI"
print(f"Loading table: (sTable} from (datawarehouse_db}")
PersonFrame0 = pd.read sql query("SELECT * FROM [Dim-BMI];", connl)
print(f"Filtering data from table: {sTable}")
filtered sql = """
    SELECT PersonID, Height, Weight, bmi, Indicator
    FROM [Dim-BMI]
    WHERE Height > 1.5 AND Indicator = 1
    ORDER BY Height, Weight;

PersonFramel = pd.read_sq1_query(fi1tered_sq1, connl)
DimPerson = PersonFrame1.set index(["PersonID"], inplace=False)
sTableHorizontal = "Dim-BMI-Horizontal"
print(f"Storing table: (sTableHorizontal} into (datamart db}")
DimPerson.to sql(sTab1eHorizonta1, conn2, if exists="replace")
print(f"Loading table: (sTableHorizontal} from (datamart_db}")
PersonFrame2 = pd.read sql query("SELECT * FROM [Dim-BMI-Horizontal];", conn2)
print(f"Full Data Set (Rows, Columns): {PersonFrame0.shape}")
print(fi'Horizontal Data Set (Rows, Columns): (PersonFrame2.shape}")
connl.close()
conn2.close()
```

Output:-

```
========================================================= RESTART: F:\DS_Practicals_Msc-it\Practi
Loading table: Dim-BMI from F:\DS_Practicals_Msc-it\Practical no 8\datawarehouse.db
Filtering data from table: Dim-BMI
Storing table: Dim-BMI-Horizontal into F:\DS_Practicals_Msc-it\Practical no 8\datamart.db
Loading table: Dim-BMI-Horizontal from F:\DS_Practicals_Msc-it\Practical no 8\datamart.db
Full Data Set (Rows, Columns): (1080, 5)
Horizontal Data Set (Rows, Columns): (194, 5)
```

2) VerticaI Style

Code:-

```
import pandas as pd
import sqlite3 as sq
datawarehouse_db = "F:\\DS_Practicals_Msc-it\\Practical no 8\\datawarehouse.db"
datamart_db = "F:\\DS_Practicals_Msc-it\\Practical no 8\\datamart.db"
connl = sq.connect(datawarehouse_db)
conn2 = sq.connect(datamart_db)
PersonFrame0 = pd.read_sql_query("SELECT * FROM [Dim-BMI];", connl)
PersonFramel = pd.read_sql_query(

    SELECT Height, Weight, Indicator
    FROM [Dim-BMI];

  connl,
)DimPerson = PersonFrame1.set_index(["Indicator”], inplace=False)
vertical_table = "Dim-BMI-Vertical"
DimPerson.to_sql(vertical_table, conn2, if_exists="replace")
PersonFrame2 = pd.read_sql_query("SELECT * FROM [Dim-BMI-Vertical];", conn2)
print(f"Full Data Set (Rows, Columns): (PersonFrame0.shape}")
print(fi'Vertical Data Set (Rows, Columns): (PersonFrame2.shape}")
connl.close()
conn2.close()
```

Output:-

```
>>
   ========================================================
   Full Data Set (Rows, Columns): (1080, 5)
   Vertical Data Set (Rows, Columns): (1080, 3)
>> D
```

3) Island Style

Code:-

```
import pandas as pd
import sqlite3 as sq
datawarehouse_db = "F:\\DS_Practicals_Msc-it\\Practical no 8\\datawarehouse.db"
datamart_db = "F:\\DS_Practica1s_Msc-it\\Practica1 no 8\\datamart.db"
connl = sq.connect(datawarehouse_db)
conn2 = sq.connect(datamart_db)
PersonFrame0 = pd.read_sql_query("SELECT * FROM [Dim-BMI];", connl)
query = """
    SELECT Height, Weight, Indicator
    FROM [Dim-BMI]
    WHERE Indicator > 2
    ORDER BY Height, Weight;

PersonFramel = pd.read_sql_query(query, connl)
DimPerson = PersonFrame1.set_index("Indicator", inplace=Fa1se)
vertical_table = "Dim-BMI-Vertical"
DimPerson.to_sql(vertical_table, conn2, if_exists="replace")
PersonFrame2 = pd.read_sq1_query("SELECT * FROM [Dim-BMI-Vertical];", conn2)
```

```
print(fi'Origina1 Data Set (Rows, Columns): (PersonFrame0.shape}")
print(fi'Vertica1 Data Set (Rows, Columns): (PersonFrame2.shape}")
connl.close()
conn2.close()
```

OUtpUt:-

```
==============================================
Original Data Set (Rows, Columns): (1080, 5)
Vertical Data Set (Rows, Columns): (771, 3)
>>
```

4) Secure Vault Style

Code:-

```
import pandas as pd
import sqlite3 as sq
datawarehouse_db = "F:\\DS_Practicals_Msc-it\\Practical no 8\\datawarehouse.db"
datamart_db = "F:\\DS_Practica1s_Msc-it\\Practica1 no 8\\datamart.db"
connl = sq.connect(datawarehouse_db)
conn2 = sq.connect(datamart_db)
PersonFrame0 = pd.read_sql_query("SELECT * FROM [Dim-BMI];", connl)
query = """
    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;

PersonFramel = pd.read_sq1_query(query, connl)
DimPerson = PersonFrame1.set_index("Indicator", inp1ace=Fa1se)
secure_table = "Dim-BMI-Secure"
DimPerson.to_sql(secure_table, conn2, if_exists="replace")
filtered_query = "SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
PersonFrame2 = pd.read_sql_query(filtered_query, conn2)
print(fi'Origina1 Data Set (Rows, Columns): (PersonFrame0.shape}")
print(f'Filtered Data Set (Rows, Columns): (PersonFrame2.shape}")
print("Only Sam Data:\n", PersonFrame2.head())
connl.close()
conn2.close()
```

Output:-

```
==============================================
Original Data Set (Rows, Columns): (1080, 5)
Filtered Data Set (Rows, Columns): (692, 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
```

5) Association Rule Mining

Code:-

```
import pandas as pd
from mlxtend.frequent    atterns import apriori, association  rules
InputFilePath = 'F:\\DS  Practicals  Msc-it\\Practica1 no 8\\Online-Retail-Billboard.xlsx'
df=  pd.read_exce1(InputFilePath)
print(IData Shape:  (df.shape}')
df['Description'] = df['Description'].str.strip()
df.dropna(subset=['InvoiceNo'], inp1ace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype(str)
df=  df[  df['InvoiceNo'].str.contains('C')]
def encode  units(x):
      return True if x > 0 else False
basket  france = (df[df['Country'] == "France"]
                          .groupby(['InvoiceNo', 'Description'])['Quantity']
                          .sum().unstack().reset  index().fillna(0).set  index('InvoiceNo'))
basket  france = basket  france.apply(lambda x: x > 0)
if'POSTAGE' in basket  france.columns:
      basket  france.drop('POSTAGE', axis=1, inplace=True)
frequent_itemsets_fr = apriori(basket_france, min_support=0.07, use_co1names=True)
rules fr          association rules(frequent itemsets fr,    metric="lift”,     min threshold=1,
num_itemsets=2)
print(rules fr[(rules fr['lift'] >= 6) & (rules fr['confidence'] >= 0.8)])
product 1 = 'ALARM CLOCK BAKELIKE GREEN'
product 2 = 'ALARM CLOCK BAKELIKE RED'
print(fi'(product_1}: (basket france[product 1].sum()}")
print(I"(product_2}:  (basket_france[product_2].sum()}")
basket  germany = (df[df['Country'] == "Germany"]
                           .groupby(['InvoiceNo', 'Description'])['Quantity']
                           .sum().unstack().reset index().fillna(0).set index('InvoiceNo'))
basket  germany = basket  germany.apply(lambda x: x > 0)
if'POSTAGE' in basket  germany.columns:
      basket  germany.drop('POSTAGE', axis=1, inplace=True)
frequent_itemsets_de = apriori(basket_germany, min_support=0.05, use_co1names=True)
rules de          association rules(frequent itemsets de,    metric="lift",     min threshold=1,
num_itemsets=2)
print(rules de[(rules de['lift'] >= 4) & (rules de['confidence'] >= 0.5)])
print(”Done")
```

OUtput:-

```
================================================ RESTART: F:\DS_Practica
Data Shape: (541909, 8)
                                        antecedents  ... kulczynski
2                        (ALARM CLOCK BAKELIKE GREEN)  ...   0.826814
3                          (ALARM CLOCK BAKELIKE RED)  ...   0.826814
16                    (SET/6 RED SPOTTY PAPER PLATES)  ...   0.784615
18                    (SET/6 RED SPOTTY PAPER PLATES)  ...   0.924444
19                      (SET/6 RED SPOTTY PAPER CUPS)  ...   0.924444
20  (SET/6 RED SPOTTY PAPER PLATES, SET/6 RED SPOT...  ...   0.781250
21  (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...  ...   0.848611
22  (SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...  ...   0.877500

[8 rows x 14 columns]
ALARM CLOCK BAKELIKE GREEN: 38
ALARM CLOCK BAKELIKE RED: 37
                        antecedents  ... kulczynski
0   (PLASTERS IN TIN CIRCUS PARADE)  ...   0.538485
6        (PLASTERS IN TIN SPACEBOY)  ...   0.507937
11    (RED RETROSPOT CHARLOTTE BAG)  ...   0.654634

[3 rows x 14 columns]
Done
```

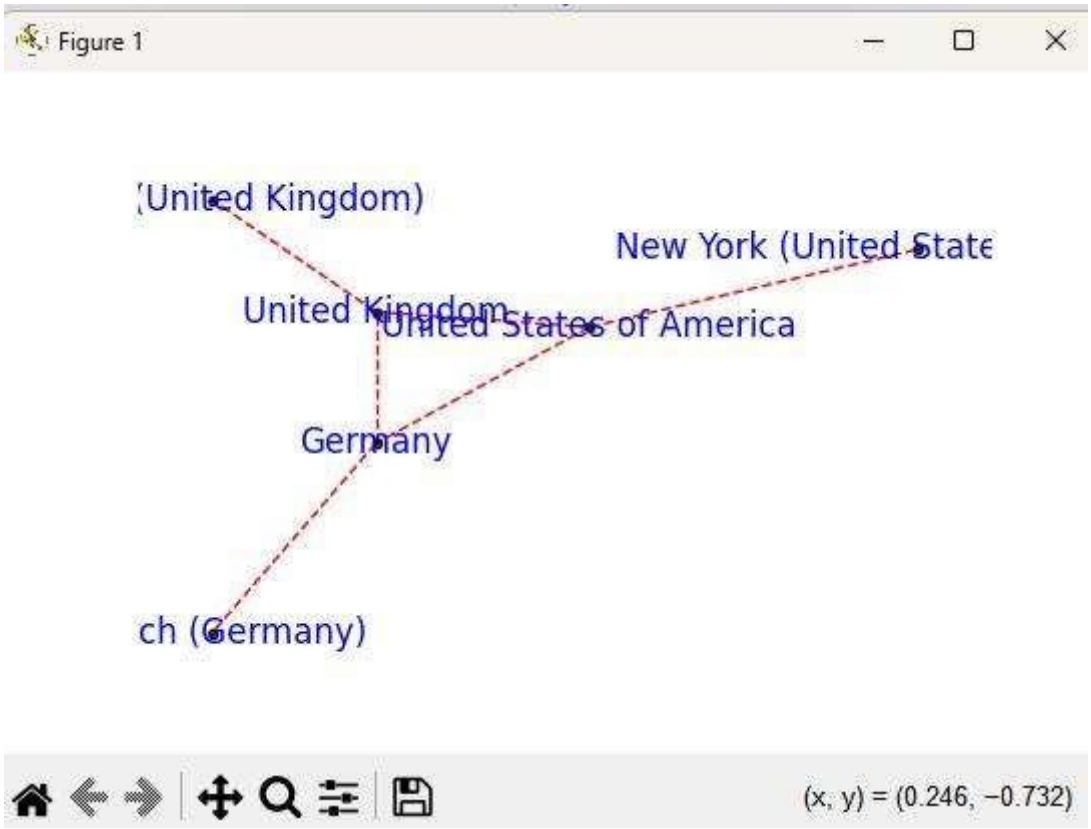
6) Create a Network Routing Diagram

Code:-

```
import pandas as pd
import networkx as nx
import matp1ot1ib.pyplot as plt
sInputFileName 'F:\\DS Practicals Msc-it\\Practical no 8\\Assess-Network-Routing-Company.csv'
sOutputFileNamel 'F:\\DS Practicals Msc-it\\Practica1 no 8\\Organise-Network-Routing-Company.gm1'
sOutputFileName2 = 'F:\\DS Practicals Msc-it\\Practica1 no 8\\Organise-Network-Routing-Company.png'
sFileName ='F:\\DS Practicals Msc-it\\Practica1 no 8\\Assess-Network-Routing-Company.csv'
print(fi'Loading: (sFileName}")
CompanyData = pd.read_csv(sFileName, header=0, 1ow_memory=False, encoding="latin-1")
print(CompanyData.head())
print(f"Data Shape: (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]
        Nodel = CompanyData['Company_Country_Name'][j]
        ifNode0 != Nodel:
            G.add edge(Node0, Nodel)
for i in range(CompanyData.shape[0]):
    Node0 = CompanyData['Company Country Name'][i]
    Nodel I"(CompanyData['Company Place Name'][i]} ((CompanyData['Company Country Name'][i]})"
    if Node0 != Nodel:
        G.add edge(Node0, Nodel)
print(f"Nodes: {G.number_of_nodes()}")
print(fi'Edges: (G.number of edges()}")
sFileName = f'(sOutputFileNamel}”
print(f"Storing GML file: (sFileName}")
nx.write gml(G, sFileName)
sFileName = I"{sOutputFileName2}"
print(fi'Storing Graph Image: (sFileName}")
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 co1or='r', arrows=Fa1se, sty1e='dashed')
nx.draw networkx labels(G, pos, font size=12, font family='sans-serif, font co1or='b')
plt.axis('off)
plt.savefig(sFileName, dpi=600)
p1t.show()
print(”Done")
```

Output:-

```
Loading: F:\DS_Practicals_Msc-it\Practical no 8\Assess-Network-Routing-Company.csv
  Company_Country_Code  ...      Company_Country_Name
0                   US  ...  United States of America
1                   US  ...  United States of America
2                   US  ...  United States of America
3                   US  ...  United States of America
4                   US  ...  United States of America

[5 rows x 5 columns]
Data Shape: (150, 5)
Nodes: 6
Edges: 6
Storing GML file: F:\DS_Practicals_Msc-it\Practical no 8\Organise-Network-Routing-Company.gml
Storing Graph Image: F:\DS_Practicals_Msc-it\Practical no 8\Organise-Network-Routing-Company.png
```



7) Picking Content for Billboards

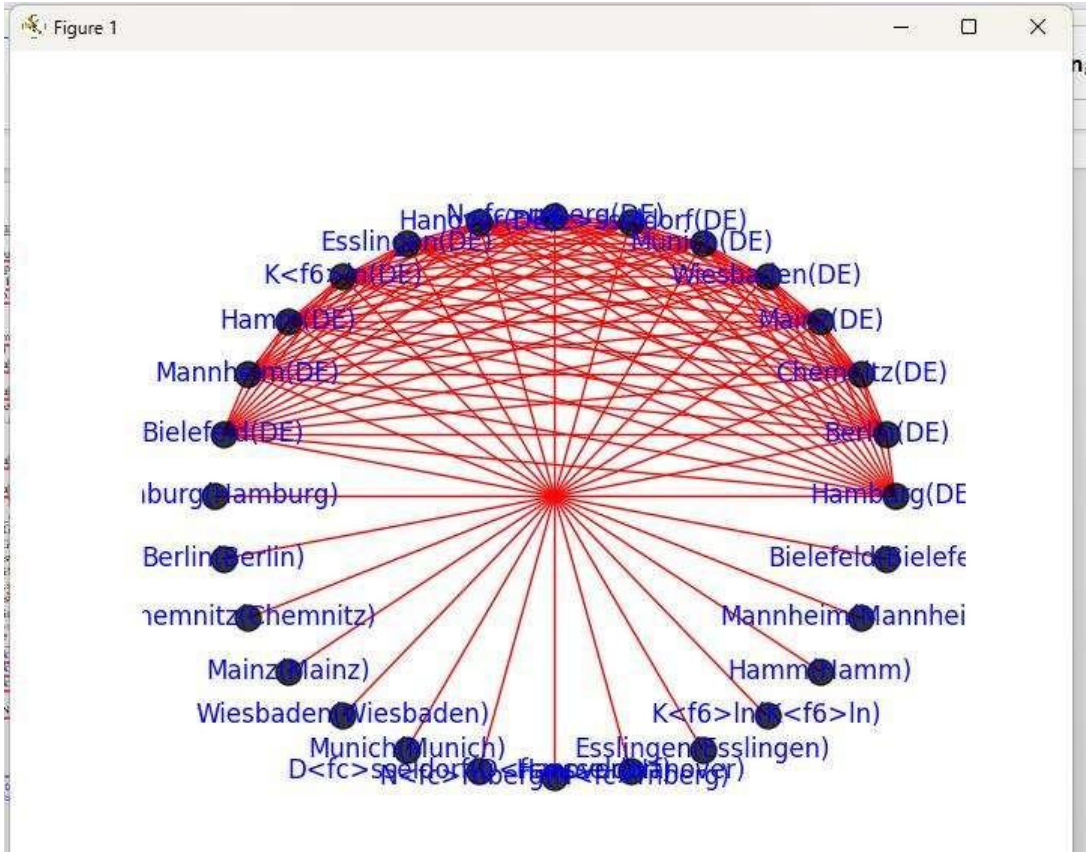
Code:-

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
sInputFileName = 'F:\\DS_Practicals_Msc-it\\Practical no 8\\Assess-DE-Billboard-Visitor.csv'
sOutputFileNamel = 'F:\\DS_Practica1s_Msc-it\\Practica1 no 8\\Organise-Billboards.gm1'
sOutputFileName2 = 'F:\\DS_Practicals_Msc-it\\Practica1 no 8\\Organise-Billboards.png'
sFileName = 'F:\\DS_Practica1s_Msc-it\\Practica1no 8\\Assess-DE-Billboard-Visitor.csv'
print(f'Loading: (sFileName}")
BillboardDataRaw = pd.read_csv(sFileName, header=0, low_memo=False, encoding="latin-
»)
```

```
print(Bi11boardDataRaw.head())
print(fi'Data Shape: (Bi11boardDataRaw.shape}")
BillboardData = BillboardDataRaw
ssample = list(np.random.choice(Bi11boardData.shape[0], 20))
G = nx.Graph()
for i in ssample:
    for j in ssample:
        Node0
fi'(BillboardData['BillboardPlaceName'][i]}((BillboardData['Bi11boardCountry'][i]})"
        Nodel
I"(BillboardData['BillboardPlaceName'][j]}((BillboardData['Bi11boardCountry'][i]})"
        ifNode0 != Nodel:
            G.add edge(Node0, Nodel)
for i in ssample:
    Node0
I'(BillboardData['BillboardPlaceName'][i]}((BillboardData['VisitorP1aceName'][i]})"
    Nodel
fi'(BillboardData['BillboardPlaceName'][i]}((Bi1lboardData['VisitorCountry'][i]})"
    ifNode0 != Nodel:
        G.add_edge(Node0, Nodel)
print(fi'Nodes: (G.number of nodes()}")
print(f"Edges: {G.number_of_edges()}")
sFileName = fi'(sOutputFileNamel}”
print(f"Storing GML file: (sFileName}")
nx.write gm1(G, sFileName)
sFileName = I'(sOutputFileName2}”
print(f"Storing Graph Image: (sFileName}")
plt.figure(figsize=(15, 15))
pos = nx.circu1ar_layout(G, dim=2)
nx.draw networkx nodes(G, pos, node colom'k', node size=150, alpha=0.8)
nx.draw networkx edges(G, pos, edge colom'r', arrows=False, style='solid')
nx.draw networkx labels(G, pos, font size=12, font fami1y='sans-serif, font colom'b')
plt.axis('off)
plt.savefig(sFileName, dpi=600)
p1t.show()
print("Done")
```

Output:-

```
================================================ RESTART: F:\DS_Practicals_Msc-it\Practic
Loading: F:\DS_Practicals_Msc-it\Practical no 8\Assess-DE-Billboard-Visitor.csv
  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

[5 rows x 9 columns]
Data Shape: (181235, 9)
Nodes: 28
Edges: 105
Storing GML file: F:\DS_Practicals_Msc-it\Practical no 8\Organise-Billboards.gml
Storing Graph Image: F:\DS_Practicals_Msc-it\Practical no 8\Organise-Billboards.png
```



8) Create a Delivery Route

Code:-

```
import pandas as pd
sOutputFileName = 'F:\\DS_Practicals_Msc-it\\Practical no 8\\Organise-Routes.csv'
sFileName = 'F:\\DS_Practica1s_Msc-it\\Practica1 no 8\\Assess_Shipping_Routes.txt'
print(fi'Loading: {sFileName}")
RouteDataRaw = pd.read csv(sFi1eName, header=0, 1ow_memory=Fa1se, sep='|', encoding="1atin-1")
RouteStart = RouteDataRaw[RouteDataRaw['StartAt'] == 'WH-KA13']
RouteDistance = RouteStart[RouteStart['Cost'] == 'DistanceMiles']
RouteDistance = RouteDistance.sort_va1ues(by=['Measure'], ascending=Fa1se)
RouteMax = RouteStart["Measure"].max()
RouteMaxCost = round(((RouteMax / 1000) * 1.5 * 2), 2)
print("Maximum (£) per day:")
print(RouteMaxCost)
RouteMean = RouteStart["Measure"].mean()
RouteMeanMonth = round(((RouteMean / 1000) * 2 * 30), 6)
print("Mean per Month (Miles):")
print(RouteMeanMonth)
```

Output:-

```
========================================= RESTART: F:\DS_Practical
Loading: F:\DS_Practicals_Msc-it\Practical no 8\Assess_Shipping_Routes.txt
Maximum (£) per day:
21.82
Mean per Month (Miles):
21.56191
```

```
ID|Cost|StartAt|EndAt|Path|Measure
1|"DistanceMiles"|"HQ-KA13"|"WH-KA13"|"HQ-KA13,WH-KA13"|0.000000
2|"DistanceMeters"|"HQ-KA13"|"WH-KA13"|"HQ-KA13,WH-KA13"|0.000000
3|"Cost"|"HQ-KA13"|"WH-KA13"|"HQ-KA13,WH-KA13"|5.000000
4|"DistanceMiles"|"HQ-KA13"|"SP-KA13-7QJ"|"HQ-KA13,WH-KA13,SP-KA13-7QJ"|0.658000
5|"DistanceMeters"|"HQ-KA13"|"SP-KA13-7QJ"|"HQ-KA13,WH-KA13,SP-KA13-7QJ"|1059.000000
6|"Cost"|"HQ-KA13"|"SP-KA13-7QJ"|"HQ-KA13,WH-KA13,SP-KA13-7QJ"|10.493500
7|"DistanceMiles"|"HQ-KA13"|"SP-KA13-7QL"|"HQ-KA13,WH-KA13,SP-KA13-7QL"|1.371000
8|"DistanceMeters"|"HQ-KA13"|"SP-KA13-7QL"|"HQ-KA13,WH-KA13,SP-KA13-7QL"|2206.000000
```

9) Simple Forex Trading Planner

Code:-

```
import pandas as pd
import sqlite3 as sq
import re
sOutputFileName = 'F:\\DS_PracticaIs_Msc-it\\PracticaI no 8\\Organise-Forex.csv'
sDatabaseName  = "F:\\DS_PracticaIs_Msc-it\\PracticaI no 8\\clark.db"
sFileName = 'F:\\DS_PracticaIs_Msc-it\\PracticaI no 8\\Process_ExchangeRate.csv'
conn = sq.connect(sDatabaseName)
print(f"Loading: {sFileName)")
ForexDataRaw = pd.read_csv(sFileName, sep=",", header=0, encoding="latin-1")
print("Forex Data Sample:")
print(ForexDataRaw.head())
if ForexDataRaw.shape[1] == 4:
    ForexDataRaw.columns = ["Date", "CodeIn", "CodeOut", "Rate"]else:
    raise ValueError(f"Unexpected number of columns in {sFileName}: {ForexDataRaw.shape[1]}")
ForexDataRaw.index.names = ['RowID']
sTable = 'Forex_All'
print(f"Storing in {sDatabaseName) Table: (sTable)")
ForexDataRaw.to_sql(sTable, conn, if_exists="replace", index=True)
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(r"\s\s+", " ", sSQL)
nMoney = pd.read_sql_query(sSQL, conn)
nMoney.index.names = ['RowID']
sTable = 'MoneyData'
print(f"Storing in (sDatabaseName) Table: {sTable}")
nMoney.to_sql(sTable, conn, if_exists="replace", index=True)
ForexDay = pd.read_sql_query("SELECT Date FROM Forex_All GROUP BY Date;", conn)
I = 0
for i in range(ForexDay.shape[0]):
    sDay = str(ForexDay['Date'][i])
    sSQL = f"""
    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(r"\s\s+", " ", sSQL)
        ForexDayRate = pd.read_sql_query(sSQL, conn)

        for j in range(ForexDayRate.shape[0]):
            sBag = str(ForexDayRate['Bag']b])
            nMoney = str(round(ForexDayRate['Money'][j], 2))
            sCodeIn = ForexDayRate['PCurrency']][j]
            sCodeout = ForexDayRate['Currency']|j]
                        sSQL = f"""
            UPDATE MoneyData
            SET Date = "(sDay)", Money = {nMoney}, Currency = "(sCodeOut}"
            WHERE Bag = {sBag} AND Currency = "{sCodeIn)";

            sSQL = re.sub(r"\s\s+", " ", sSQL)
            cur = conn.cursor()
            cur.execute(sSQL)
            conn.commit()
            sSQL = f"""
            INSERT INTO TransactionData (RowID, Bag, Date, Money, Currency)
            SELECT {t) AS RowID, Bag, Date, Money, Currency
            FROM MoneyData;

            sSQL = re.sub(r"\s\s+", " ", sSQL)
            cur.execute(sSQL)
            conn.commit()
            t +-- 1
            print(f"Trade: (t}, (sDay), (sCodeOut), {nMoney}")
sSQL = "SELECT RowID, Bag, Date, Money, Currency FROM TransactionData ORDER BY RowID;"
sSQL = re.sub(r"\s\s+", " ", sSQL)
TransactionData = pd.read_sql_query(sSQL, conn)
OutputFile = f"(sOutputFileName)"
TransactionData.to_csv(OutputFile, index=False)
print("Done")
```

Output:-

```
Loading: F:\DS_Practicals_Msc-it\Practical no 8\Process_ExchangeRate.csv
Forex Data Sample:
    ï»¿Date CodeIn CodeOut        Rate
0  20170505    USD     USD    1.000000
1  20170505    USD     JPY  112.480613
2  20170505    USD     BGN    1.784326
3  20170505    USD     CZK   24.432990
4  20170505    USD     DKK    6.784874
Storing in F:\DS_Practicals_Msc-it\Practical no 8\clark.db Table: Forex_All
Storing in F:\DS_Practicals_Msc-it\Practical no 8\clark.db Table: MoneyData
Trade: 1, 20170505, GBP, 772830.95
Done
```

Pratical no 8
Aim:-Generating Data

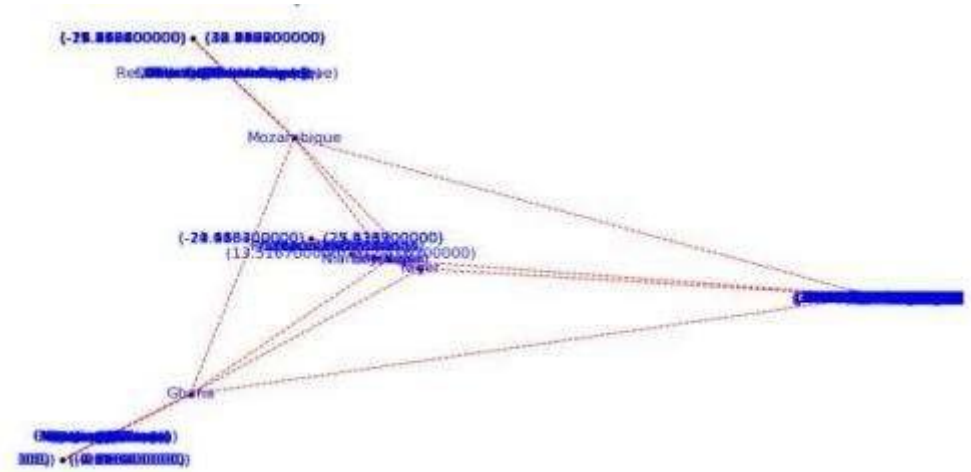
1) Vermeulen PLC

Code:-

```
import os
import pandas as pd
import networkx as nx
import matplot1ib.pyplot as plt
pd.options.mode.chained assignment = None
sInputFileName = 'F:\\DS Practicals Msc-it\\Practical no 9\\Vermu1enPLC DATA\\Assess-Network-Routing-Customer.csv'
sOutputFileNamel = 'F:\\DS_Practica1s_Msc-it\\Practical no 9\\Vermu1enPLC_DATA\\Report-Network-Routing-Customer.gml'
sOutputFileName2 = 'F:\\DS_Practica1s_Msc-it\\Practical no 9\\VermulenPLC_DATA\\Report-Network-Routing-Customer.png'
Company = '01-Vermeu1en'
sFileName = I"(sInputFileName}"
print(fi'Loading: (sFileName}")
CustomerData = pd.read_csv(sFileName, header=0, low_memory=Fa1se, encoding="latin-1").head(100)
print(f"Loaded columns: {CustomerData.columns.values}")
G = nx.Graph()
for i in range(CustomerData.shape[0]):
    NodeCountry = CustomerData['Customer Country_Name'][i]
    NodePlace = fi'(CustomerData['Customer Place Name'][i]} ((NodeCountry})"
    NodeCoords I"((CustomerData['Customer_Latitude'][i]:.9f}, {CustomerData['Customer Longitude']][i]:.9f})"
    G.add edge(NodeCountry, NodePlace)
    G.add edge(NodeP1ace, NodeCoords)
print(f"Nodes: (G.number of nodes()}")
print(f"Edges: (G.number of_edges()}")
sFileNameGML = fi'(sOutputFileNamel}"
print(f"Storing graph in GML format: {sFileNameGML}")
nx.write gm1(G, sFileNameGML)
sFileNameImg = I'{sOutputFileName2}"
print(fi'Storing graph image: (sFileNameImg}")
plt.figure(figsize=(25, 25))
pos = nx.spectral layout(G,dim=2)
nx.draw networkx nodes(G, pos, node co1om'k', node size=10, a1pha=0.8)
nx.draw_networkx_edges(G, pos, edge_co1or='r',arrows=False, style='dashed')
nx.draw networkx labels(G, pos, font size=l2,font family='sans-serif, font co1or='b')
plt.axis('off)
plt.savefig(sFileNameImg, dpi=600)
p1t.show()
print("Done")
```

Output:-

```
Loading: F:\DS_Practicals_Msc-it\Practical no 9\VermulenPLC_DATA\Assess-Network-Routing-Customer.csv
Loaded columns: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
Nodes: 205
Edges: 200
Storing graph in GML format: F:\DS_Practicals_Msc-it\Practical no 9\VermulenPLC_DATA\Report-Network-Routing-Customer.gml
Storing graph image: F:\DS_Practicals_Msc-it\Practical no 9\VermulenPLC_DATA\Report-Network-Routing-Customer.png
Done
```



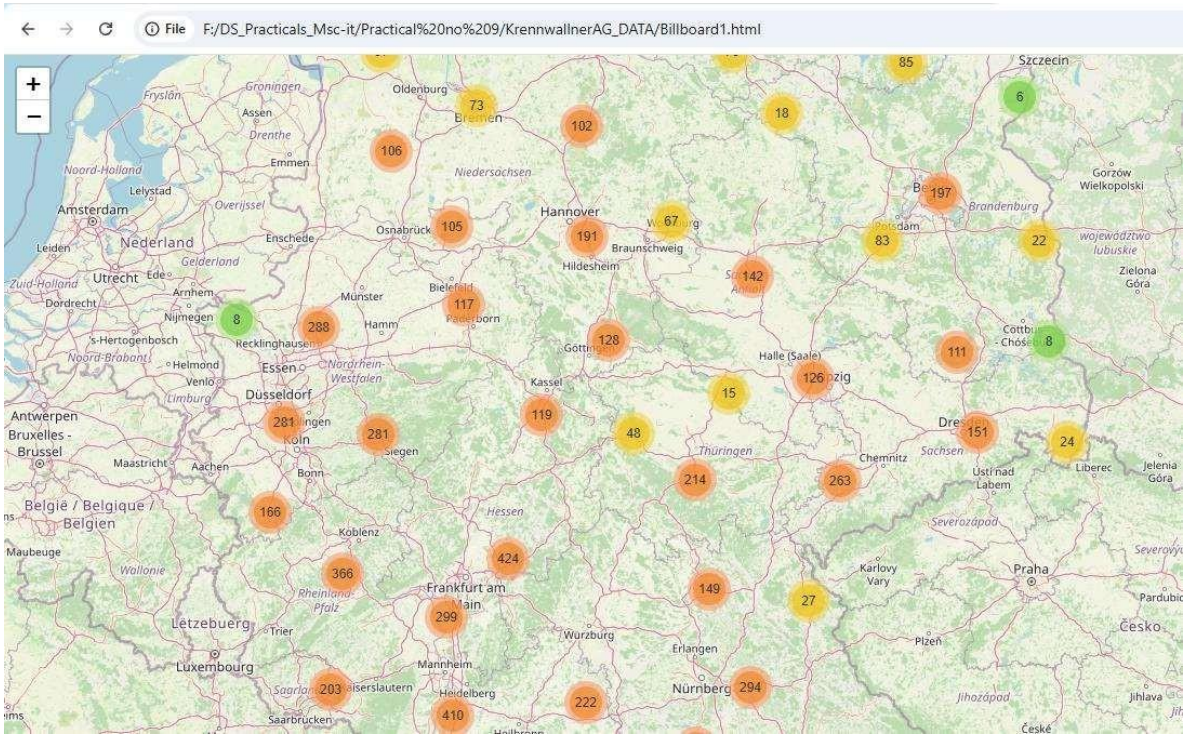
2) Krennwallner AG

Code:-

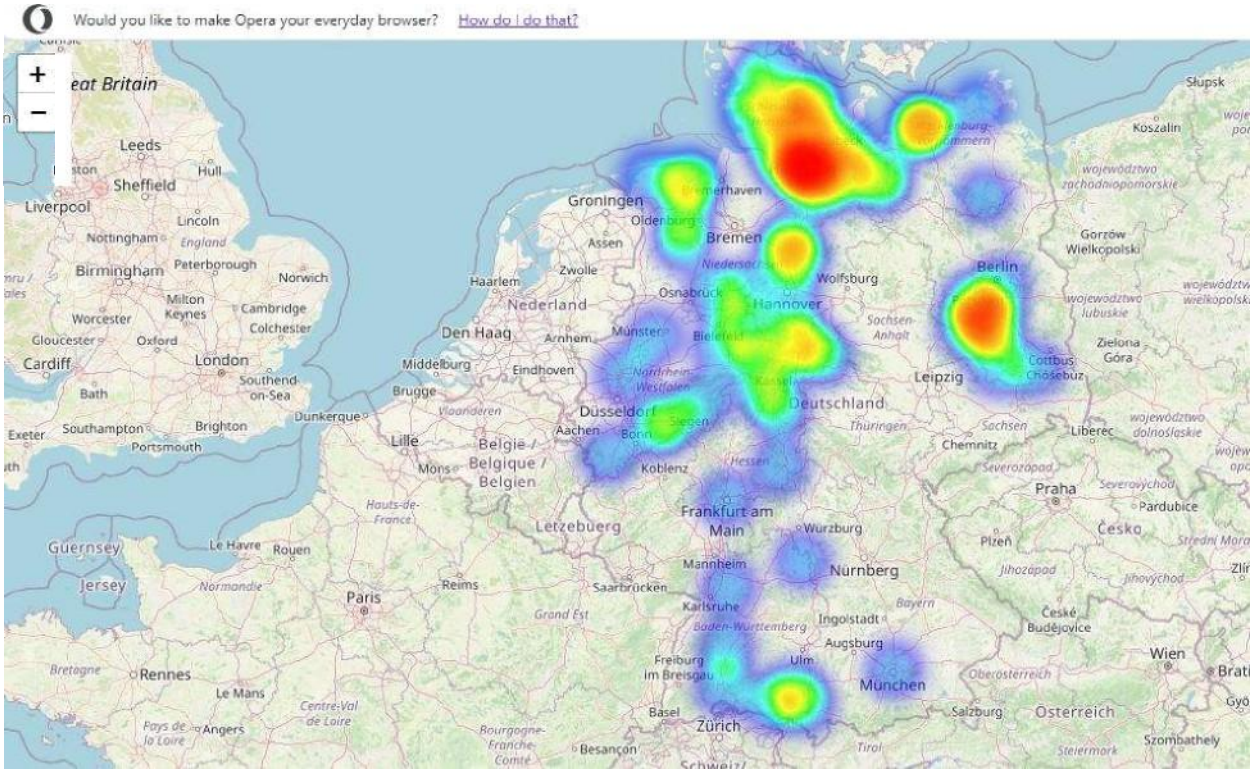
```
import os
import pandas as pd
from folium.p1ugins import FastMarkerCluster, HeatMap
from folium import Marker, Map
import webbrowser
sFileName f"F:\\DS Practicals Msc-it\\Practical no 9\\Krennwa11nerAG DATA\\Retrieve DE Billboard Locations.csv"
df= pd.read csv(sFi1eName, header=0, low memory=Fa1se, encoding="1atin-1")
df.fi11na(va1ue=0, inp1ace=True)
print(df.shape)
DataCluster = []
DataPoint = []

for i in range(df.shape[0]):
    t'v-
        sLongitude = float(df["Longitude"][i])
    except Exception:
        sLongitude = 0.0

    t'v
        sLatitude = float(df["Latitude"][i])
    except Exception:
        sLatitude = 0.0

    tv
        sDescription = f'(df['Place Name'][i]} ((df['Country'][i]})"
    except Exception:
        sDescription = 'VKHCG'
```

```
    if sLongitude != 0.0 and sLatitude != 0.0:
        DataCluster.append([sLatitude, sLongitude])
        DataPoint.append([sLatitude, sLongitude, sDescription])
pins = pd.DataFrame(DataPoint, columns=['Latitude', 'Longitude', 'Description'])
stops_mapl = Map(location=[48.1459806, 11.4985484], zoom_start=5)
FastMarkerCluster(DataCluster).add_to(stops_map1)
sFileNameHtmll = f'F:\\DS_Practicals_Msc-it\\Practical_no_9\\Krennwa11nerAG_DATA\\Billboard1.html"
stops_mapl.save(sFileNameHtmll)
webbrowser.open('file://' + os.path.realpath(sFileNameHtmll))
stops_map2 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
for _, row in pins.i1oc[: 100].iterrows():
    Marker([row["Latitude"], row["Longitude"]], popup=row["Description"]).add_to(stops_map2)
sFileNameHtml2 = f"F:\\DS_Practica1s_Msc-it\\Practical_no_9\\KrennwallnerAG_DATA\\Billboard2.html"
stops_map2.save(sFileNameHtml2)
webbrowser.open('file://' + os.path.rea1path(sFileNameHtml2))
stops_heatmap = Map(location=[48.1459806, 11.4985484], zoom_start=5)
stops_heatmap.add_chi1d(HeatMap([[row["Latitude"], row["Longitude"]] for _, row in pins.iloc[:100].iterrows()]))
sFileNameHtml3 = f"F:\\DS_Practica1s_Msc-it\\Practical_no_9\\KrennwallnerAG_DATA\\Billboard_heatmap.html"
stops_heatmap.save(sFileNameHtml3)
webbrowser.open('file://' + os.path.rea1path(sFileNameHtml3))
print("Done")
```

Output:-



O Ö file:/,','Fş'DS PraciÎcals fvlsc-İt/Practical%2Dno%209/Krenn'w'ailnerAG DATA/Billboard heatmap.html



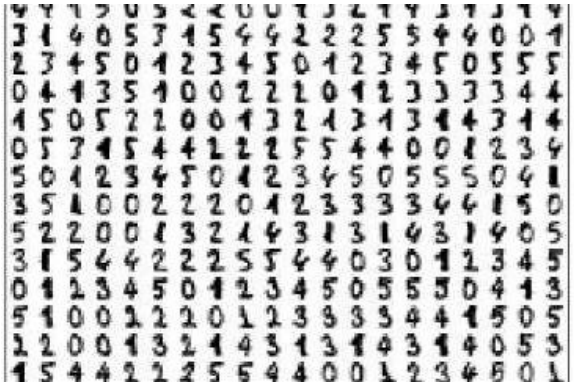
@ file:///F:y'OS Prac!icels !vł sc - it/Pi ac*tcaI %24no9ù209/KrennwallnerAG DAïA,+BiII board2.h*mI

u like to make Operz Ş'our ever'da,' browser F'o'z7 U l r4n *hat'

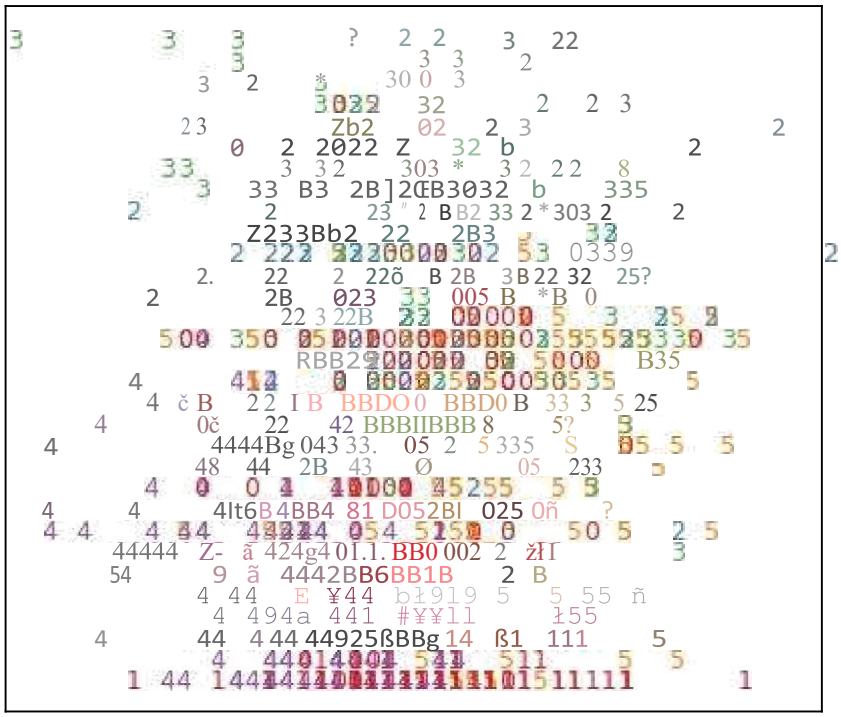
3) Hillman Ltd

Code:-

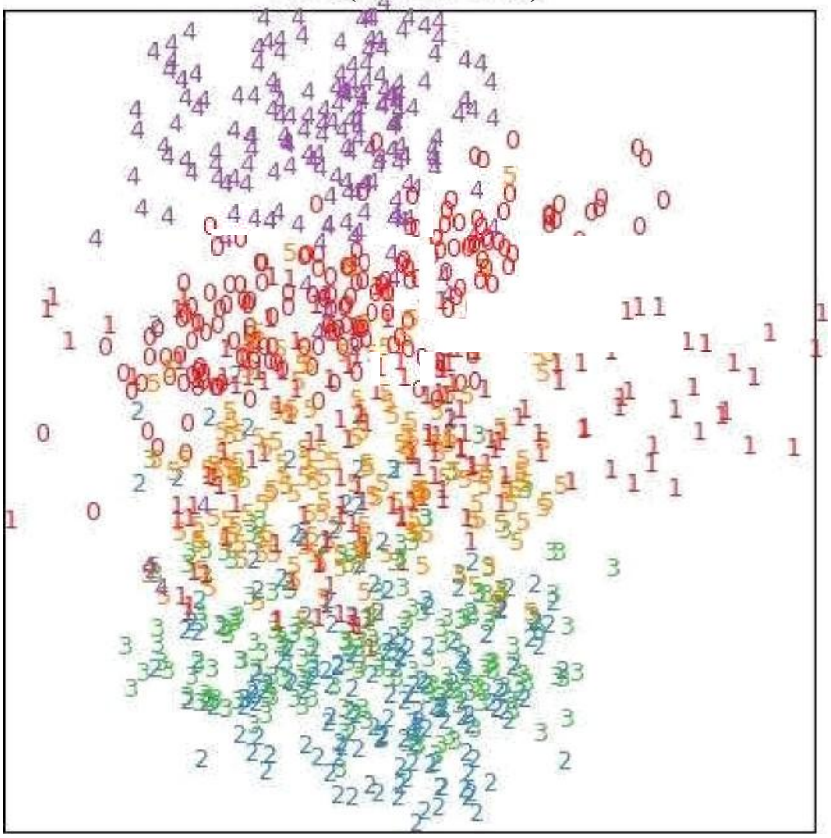
```
from time import time
import numpy as np
import matp1ot1ib.pyplot as plt
from sklearn import manifold, datasets, decomposition, discriminant analysis,
random rojection
digits = datasets.1oad_digits(n_class=6)
X, y = digits.data, digits.target
def plot embedding(X, title):
    X = (X - X.min(0)) / (X.max(0) - X.min(0))
    plt.figure(figsize=(8, 8))
    for i in range(X.shape[0]):
        plt.text(X[i, 0], X[i, 1], str(y[i]), colo 1t.cm.Set1(y[i] / 10.0), fontsize=9)
    plt.xticks([]), plt.yticks([]), plt.title(title)
plt.figure(figsize=(8, 8))
plt.imshow(
    np.concatenate([X[i].reshape(8, 8) for i in range(100)], axis=1),
    cmap lt.cm.binary,
)
plt.title("Sample digits from dataset"), p1t.axis("ofP'), plt.show()
methods = {
    "Random Projection": random rojection.SparseRandomProjection(n components=2,
random state=42),
    "PCA": decomposition.TruncatedSVD(n components=2),
    "LDA": discriminant analysis.LinearDiscriminantAnalysis(n components=2),
    "Isomap": manifold.Isomap(n_neighbors=30, n_components=2),
    "LLE": manifold.Loca11yLinearEmbedding(n neighbors=30, n components=2),
    "t-SNE": manifold.TSNE(n_components=2, init="pca", random_state=0),

for name, model in methods.items():
    print(f'Computing (name} embedding...")
    t0 = time()
    X_embedded = model.fit_transform(X, y) if name == "LDA" else model.fit_transform(X)
    plot embedding(X embedded, f'(name} (time (time() - t0:.2f}s)")
p1t.show()
```

Output;- 

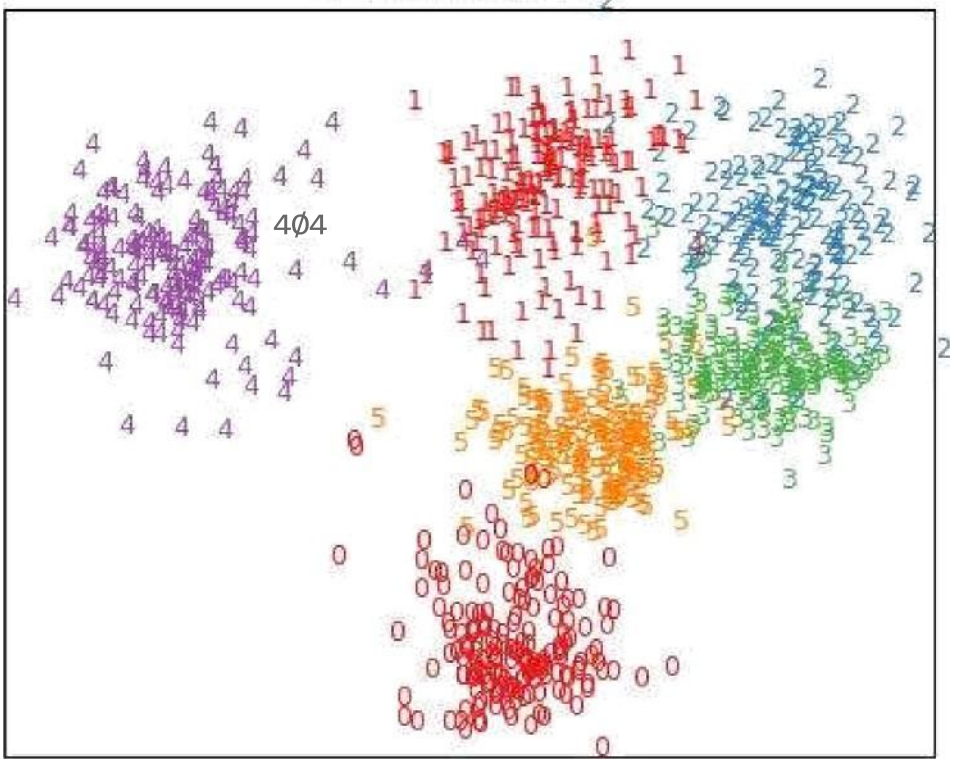
Ran m Projection (time 0.01s)



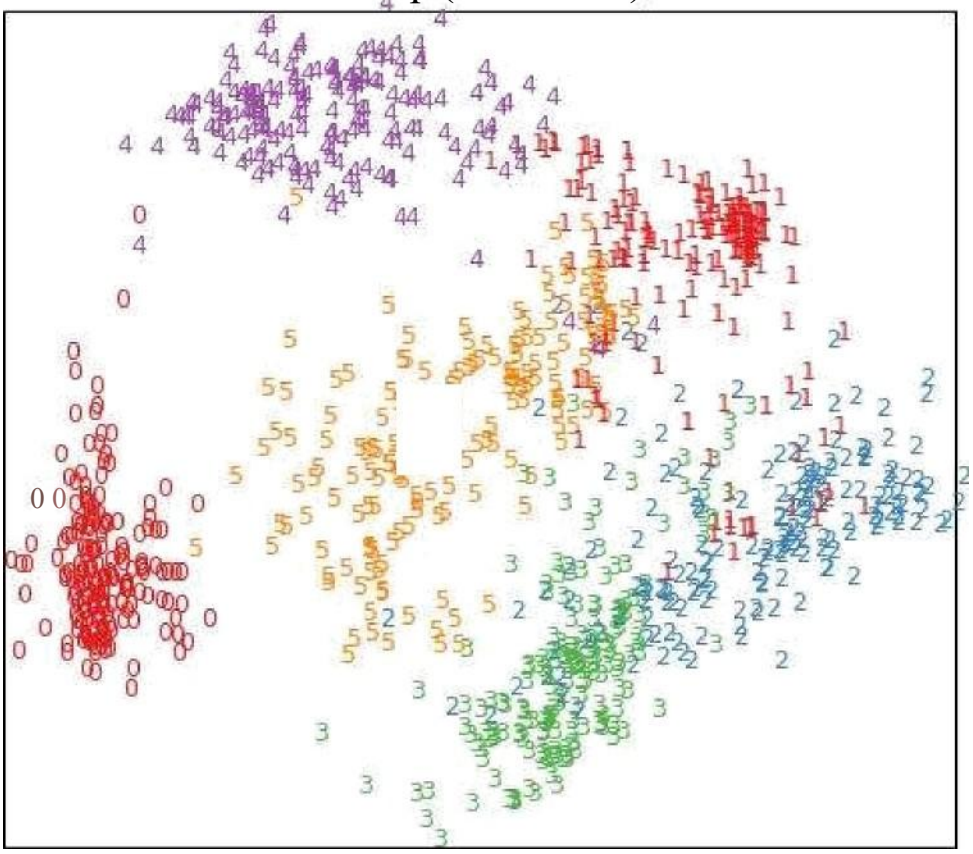
PCA (time 0.03s)



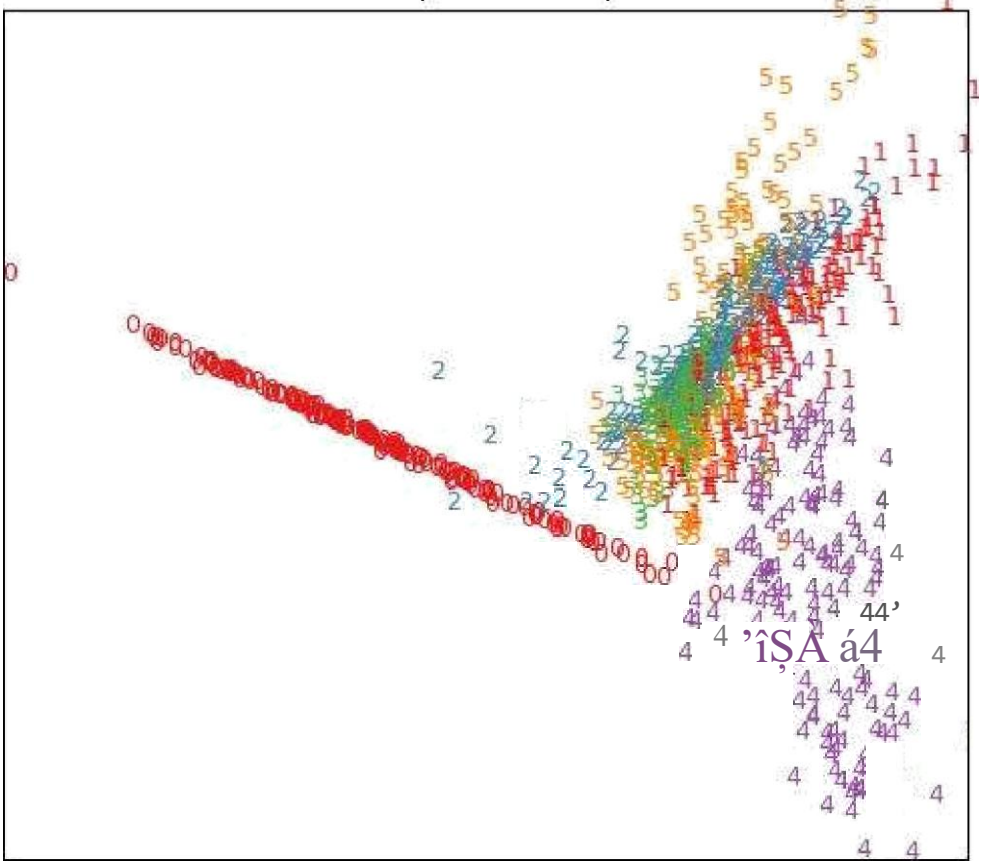
LDA (time 0.03 s)



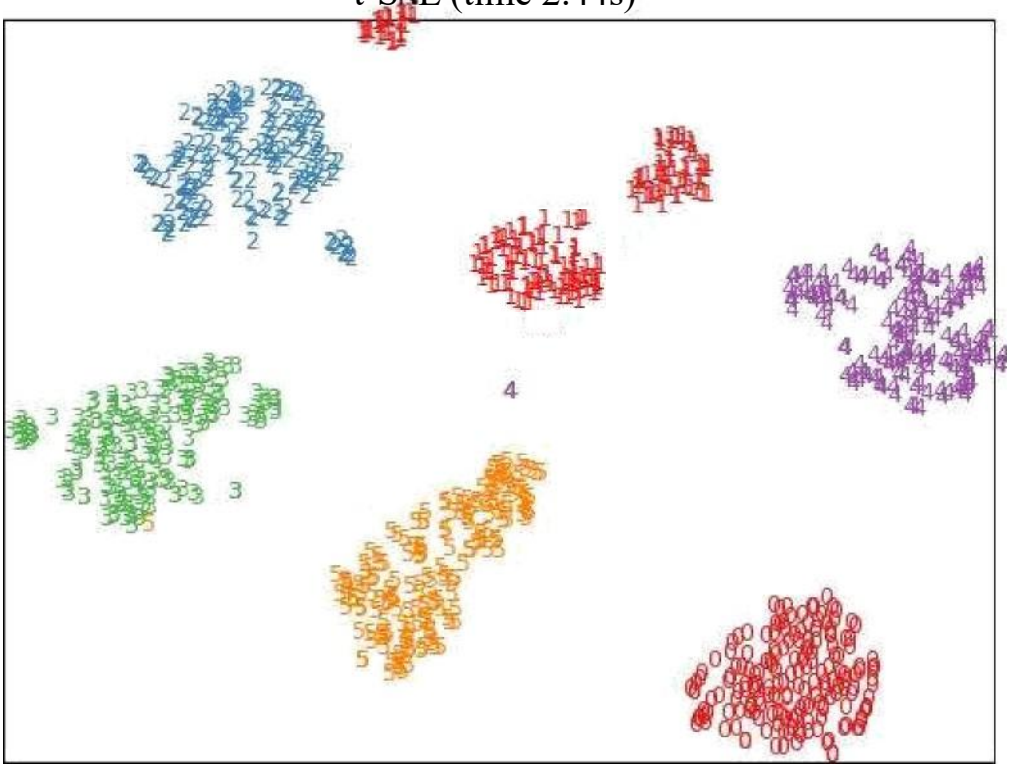
Isomap (time 3.38s)



LLE (time O.18s)



t-SNE (time 2.44s)



4) Clark Ltd

Code:-

```
import os
import pandas as pd
import sqlite3 as sq
from openpyxl import 1oad_workbook
import re
DATABASE PATH = "F:\\DS Practicals Msc-it\\Practical no 9\\C1arkLTD DATA\\clark.db"
TEMPLATE PATH = "F:\\DS Practicals Msc-it\\Practical no 9\\ClarkLTD DATA\\Balance-
Sheet-Template.xlsx"
OUTPUT PATH = "F:\\DS Practicals Msc-it\\Practical no 9\\C1arkLTD DATA"
conn = sq.connect(DATABASE PATH)
for month in range(1, 13):
    file ath fi'F:\\DS Practicals Msc-it\\Practical no
9\\Clarktltdba1ancesheet\\Ba1anceSheets{str(month).zfill(2)}.csv"
    print(f'Loading: (file ath}")
    raw data = pd.read csv(file ath, header0, encoding="1atin-1", low memo False)
    raw data.index.names = ['RowID']
    table name = 'BalanceSheets'
    if month == 1:
        raw data.to sql(table name, conn, if exists="replace")
    else:
        raw data.to sql(table name, conn, if exists="append")
sql query = """
    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
    WHERE Year IS NOT NULL
    GROUP BY Year, Quarter, Country, Company;

dates data = pd.read sql query(sql query, conn)
print("Fetched report data:", dates data.shape)
fields = 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', l],
    ['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]


for i in range(dates data.shape[0]):
    year = str(int(dates data['Year'][i]))
    quarter = str(int(dates data['Quarter'][i]))
    country = str(dates_data['Country'][i])
    company = str(dates data['Company'][i])
    company_fi1e = re.sub(r"\s+", "", dates_data['sCompanyFile'][i])
    output file = fi'(OUTPUT PATH}/Report-Balance-Sheet-(company file}.xlsx"
    workbook = load workbook(TEMPLATE PATH)
    worksheet = workbook["Balance-Sheet"]
    worksheet['D3'] = dates data['sDate'][i]
    worksheet['D5'] = dates data['sCompanyName'][i]


    for field name, cell, sign in fields.itertuples(index=False):
        field query = f"""
            SELECT SUM((field name}) AS nSumTotal
            FROM BalanceSheets
            WHERE Year=(year} AND Quarter=(quarter}
            AND Count'(country}' AND Company='(company}",

        field data = pd.read sql query(re.sub(r"\s+", " ", field query), conn)
        worksheet[cell] = field data["nSumTota1"][0] * sign
        print(f'Set (cell} to (field name} total.")
    workbook.save(output file)
    print(f"Report saved: (output file}")
```

OUTPUT:-

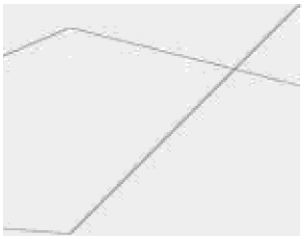
Pie Graph

Double Pie

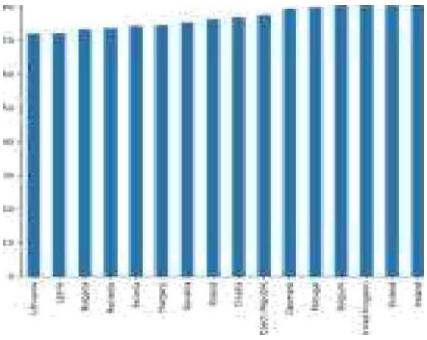
C:".VKI ICGDl-VermeulenDfi-Report Report Graph A.py

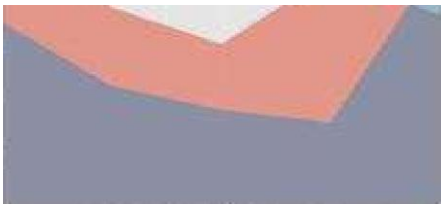
Line Graph

C:/VKI ICG/01 -Vermeulen/06-Report/Report Graph A.py



Ever C>repfz .' Horizorrta Bar C5reph





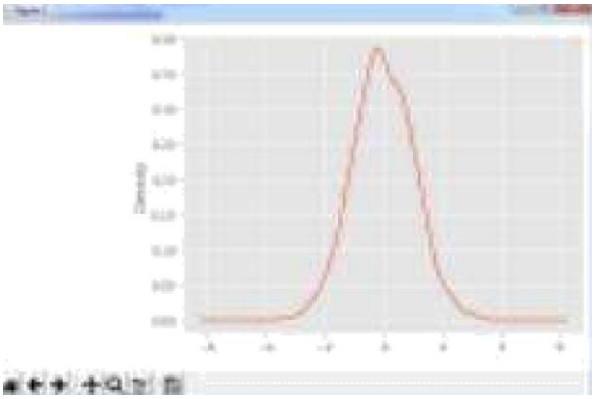
Scatter Graph : VKHCG\03-Hillman\06-Report\Report-Scatterplot-With-Encircling.

Hexbin:

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

Kernel Density Estimation (KDE) Graph

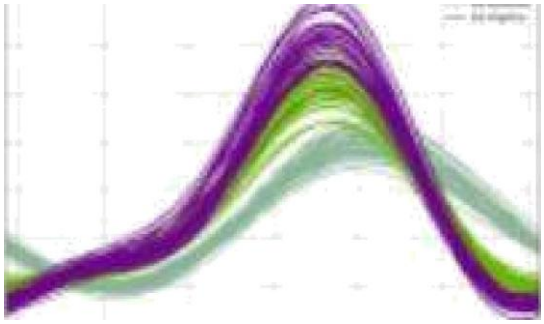
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py



Scatter Matrix Ğrapb

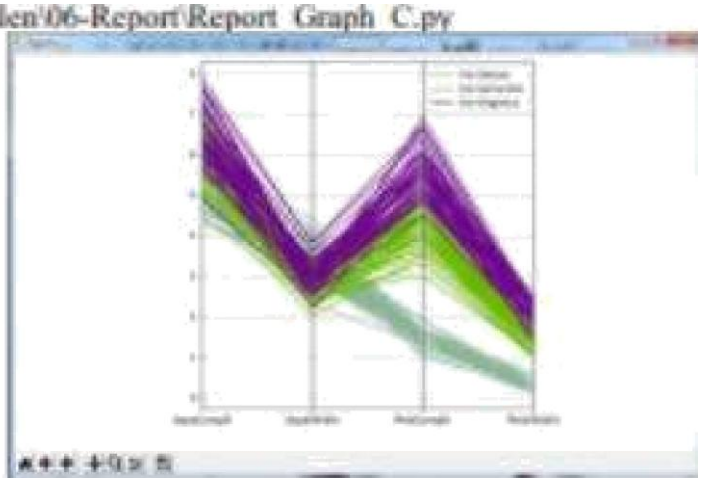
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py

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



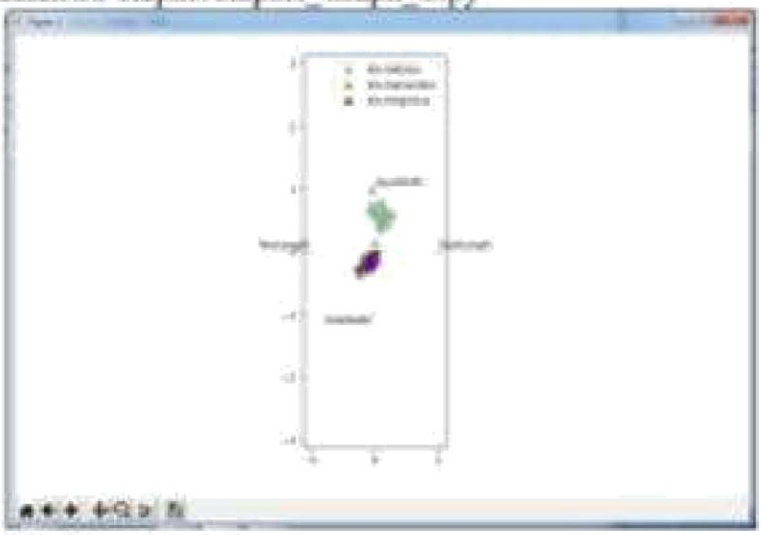
Parallel Coordinates

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



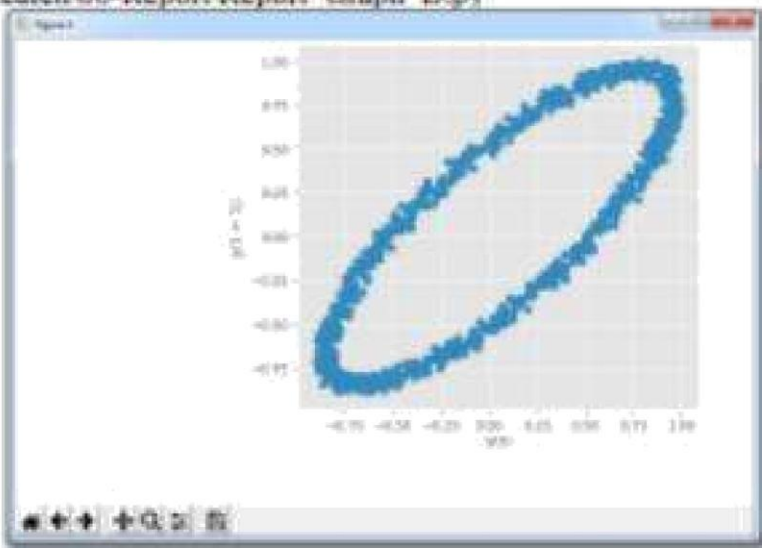
RAOVtZ k0ctluad

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

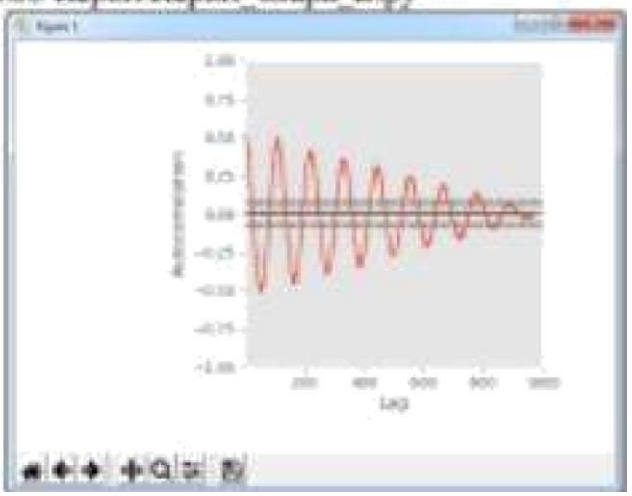


Lag Plot

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



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



3D Graphs

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

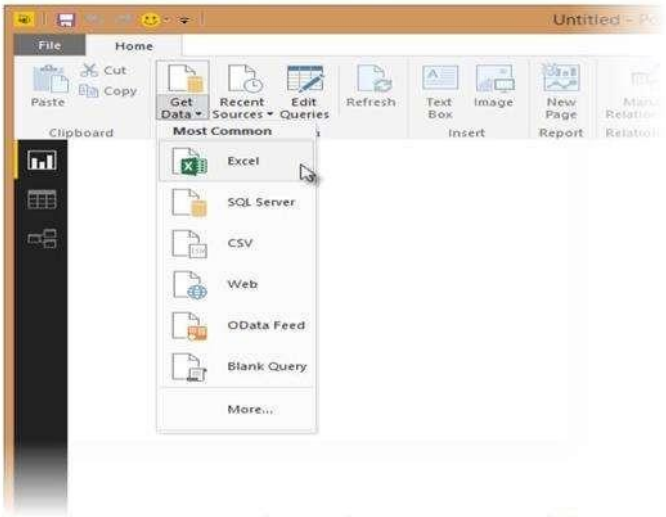
(add →import matplotlib.cm as cm & Replace : plt.cm.spectral →cm.get_cmap("Spectral") at Line 44)

Practical no 9
Aim:-Data visualisation using power Bi

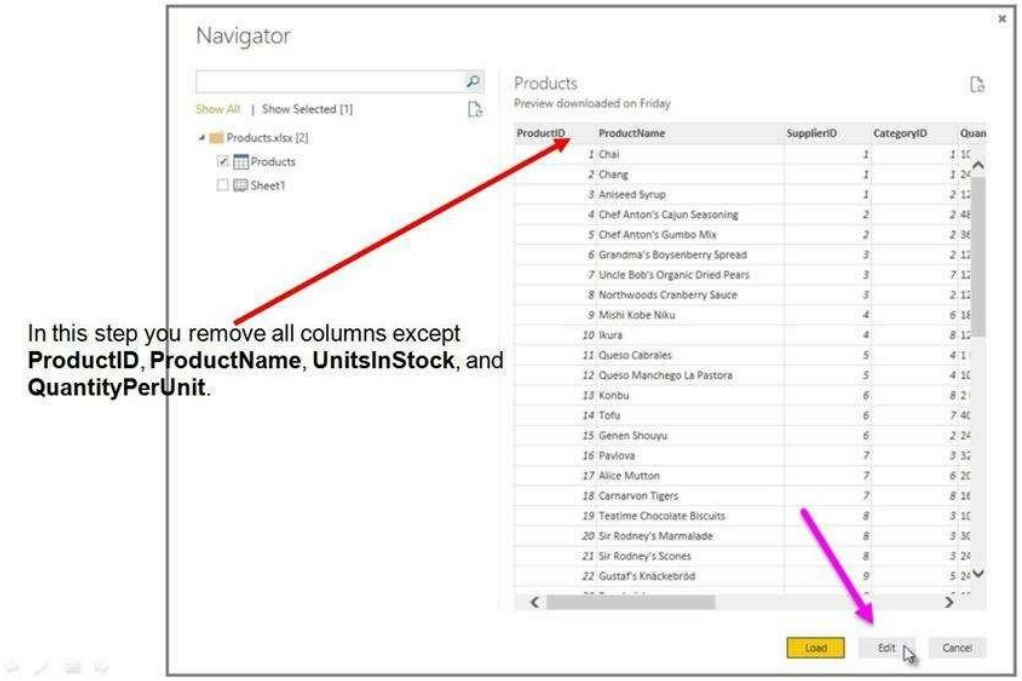
Case Study : Sales Data

Step 1: Connect to an Excel workbook

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



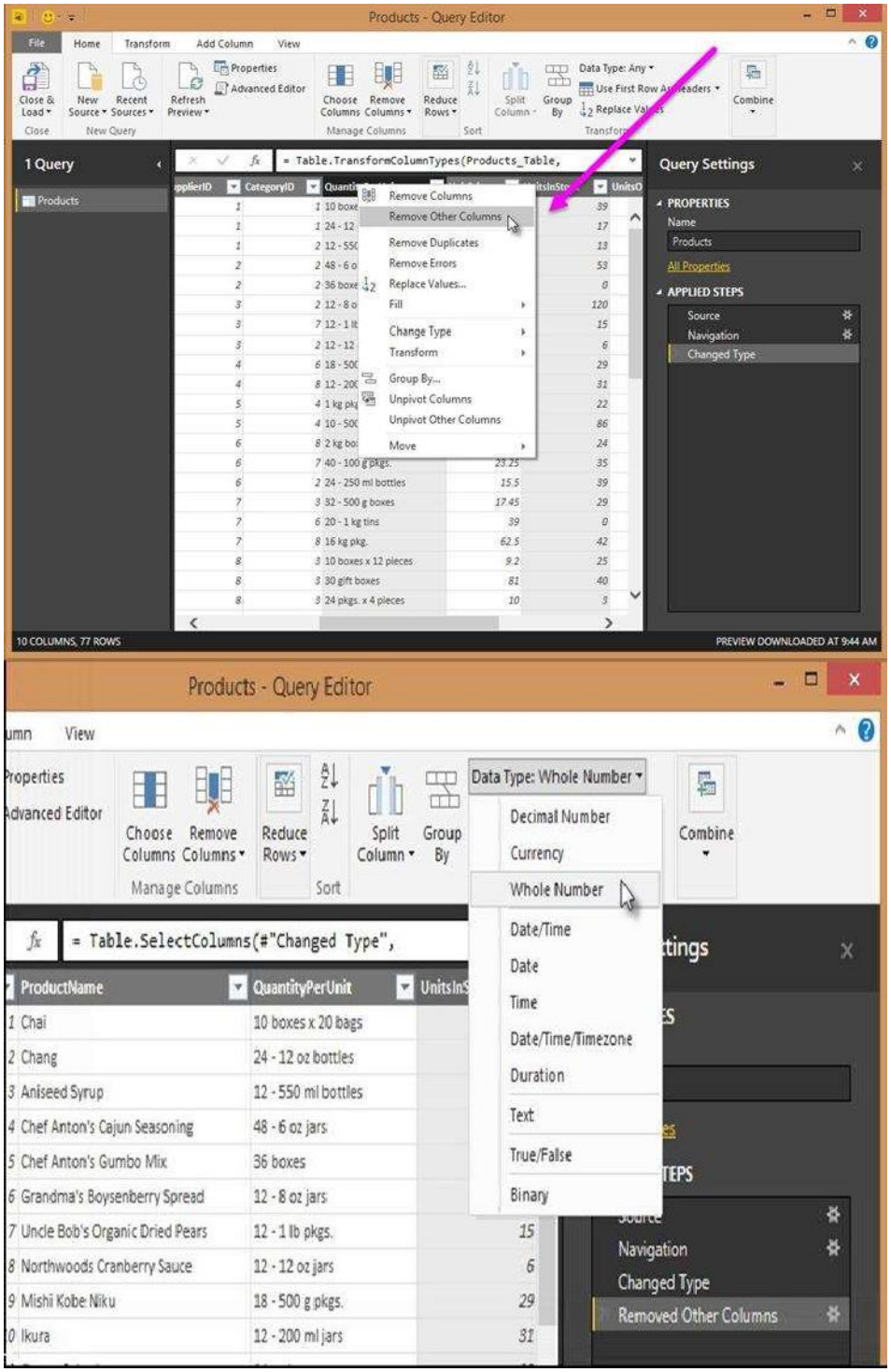
3. If you select the Get Data button directly, you can also select **File > Excel** and select **Connect**.
4. In the **Open File** dialog box, select the **Products.xlsx** file.



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

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

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



3. Change the data type of the UnitsInStock column

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

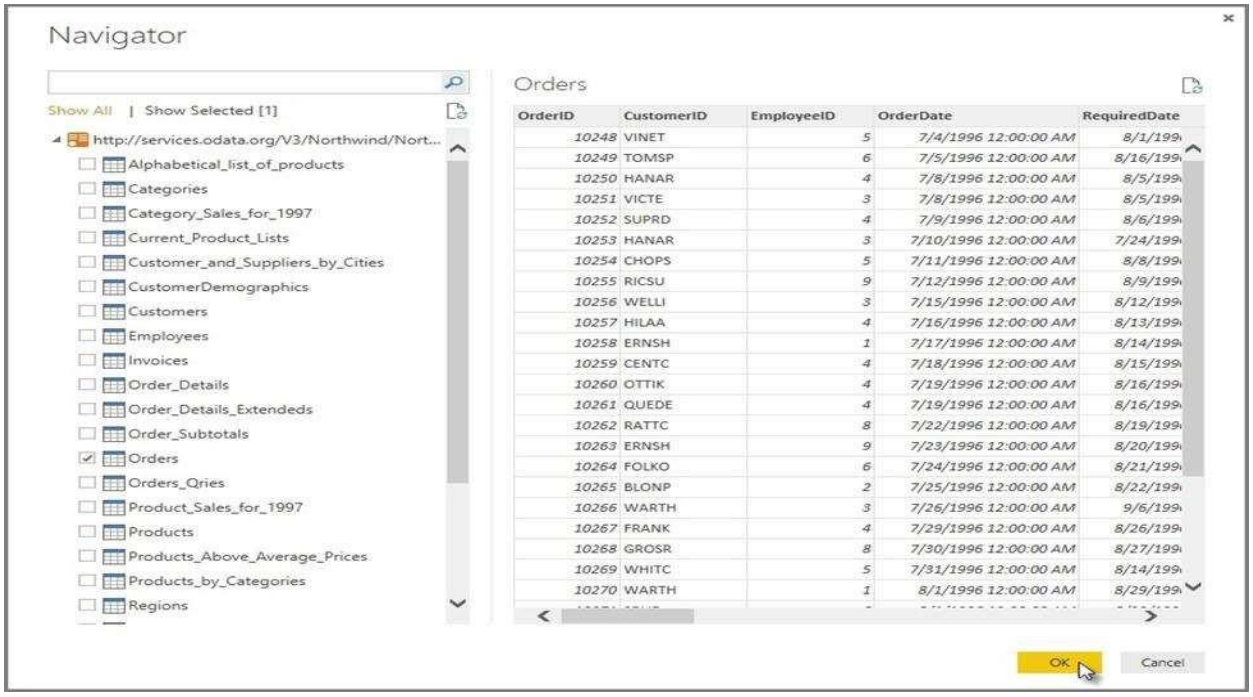
1. Select the UnitsInStock column.
2. Select the Data Type drop-down button in the Home ribbon.
3. If not already a Whole Number, select Whole Number for data type from the drop down (the Data Type: button also displays the data type for the current selection).

Task 2: Import order data from an OData feed

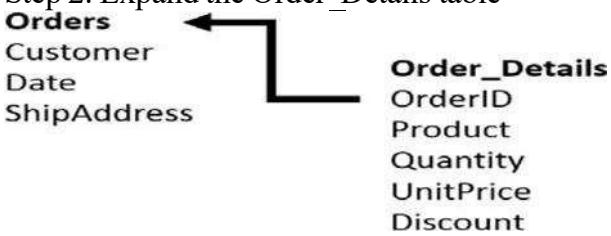
You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below:
http://services.odata.org/V3/Northwind/Northwind.svc/

Step 1: Connect to an OData feed

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



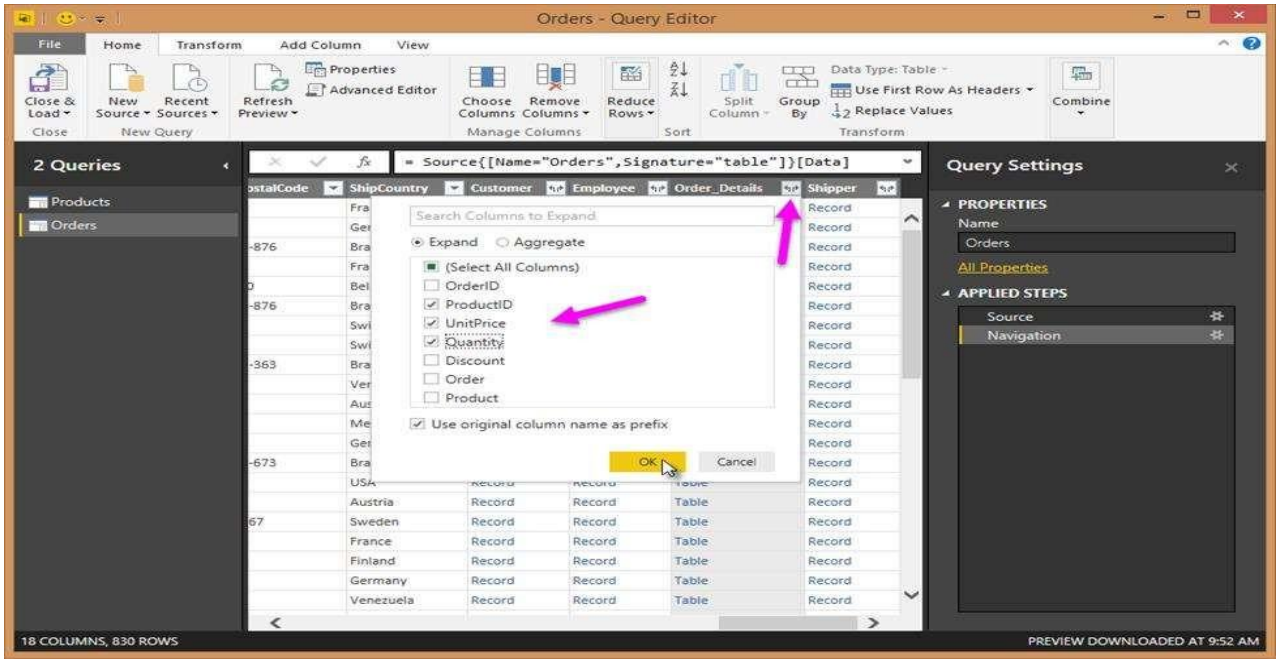
Step 2: Expand the Order_Details table



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

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

click OK.



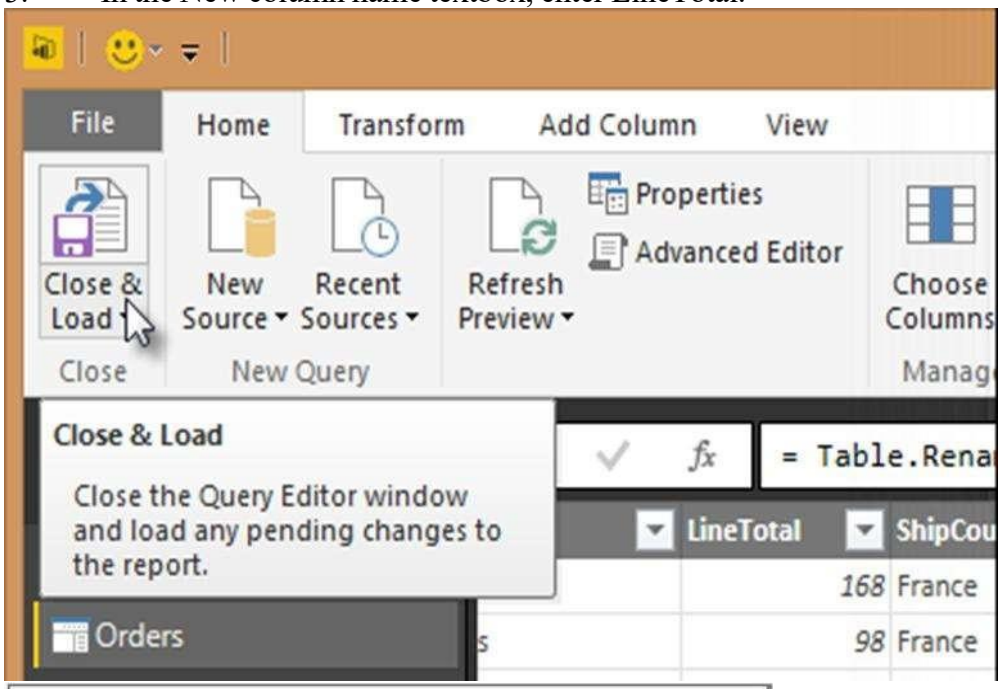
Step 3: Remove other columns to only display columns of interest
In this step you remove all columns except OrderDate, ShipCity, ShipCountry, Order Details.ProductID, Order Details.UnitPrice, and Order Details.Quantity columns. In the previous task, you used Remove Other Columns. For this task, you remove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

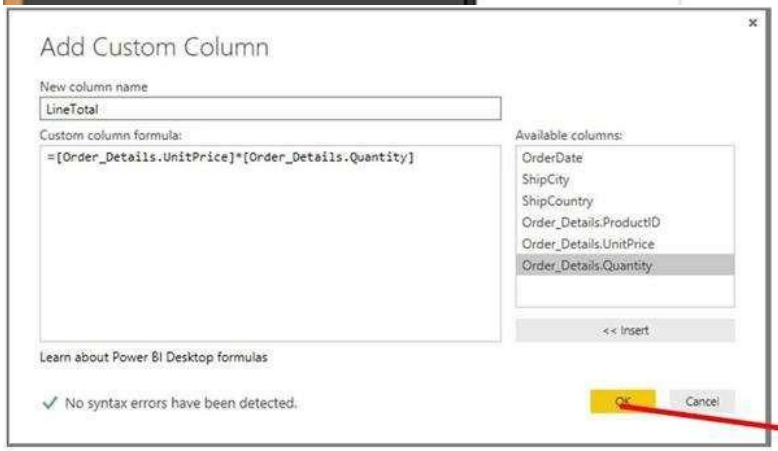
a) Click the first column (OrderID).
b) Shift+Click the last column (Shipper).
c) Now that all columns are selected, use Ctrl+Click to unselect the following columns: OrderDate, ShipCity, ShipCountry, Order Details.ProductlD, Order Details.UnitPrice, and Order_Detai1s.Quantity.

Now that only the columns we want to remove are selected, right-click on any selected column header and click Remove Columns.
Step 4: Calculate the line total for each Order_Details row
Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order Details row.
Calculate the line total for each Order Details row:

1. In the Add Column ribbon tab, click Add Custom Column.

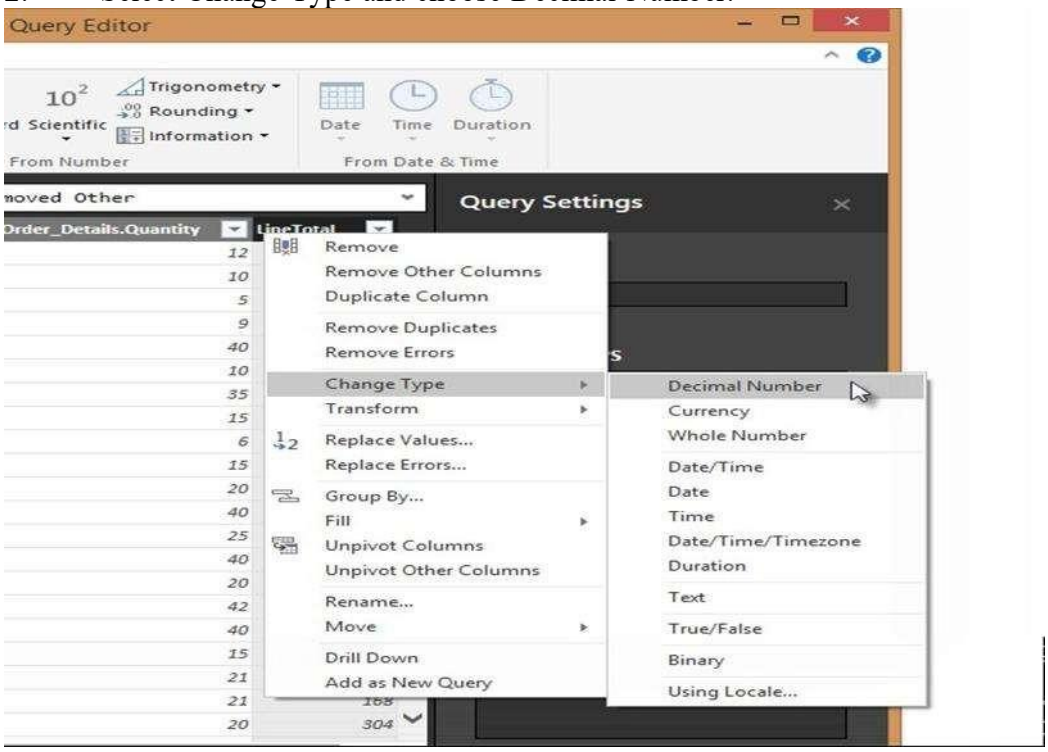
2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order Details.UnitPrice] * [Order Details.Quantity].
3. In the New column name textbox, enter LineTotal.





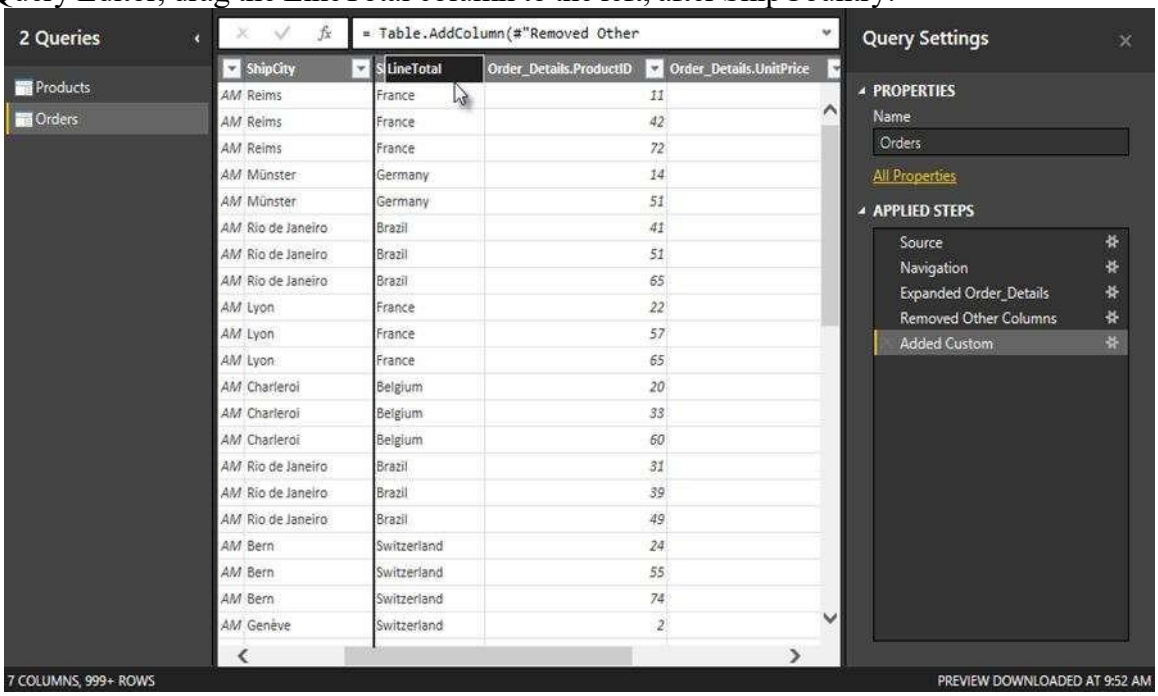
Click **OK**.

Step 5: Set the datatype of the LineTotal field
Right click the LineTotal column.
2. Select Change Type and choose Decimal Number.

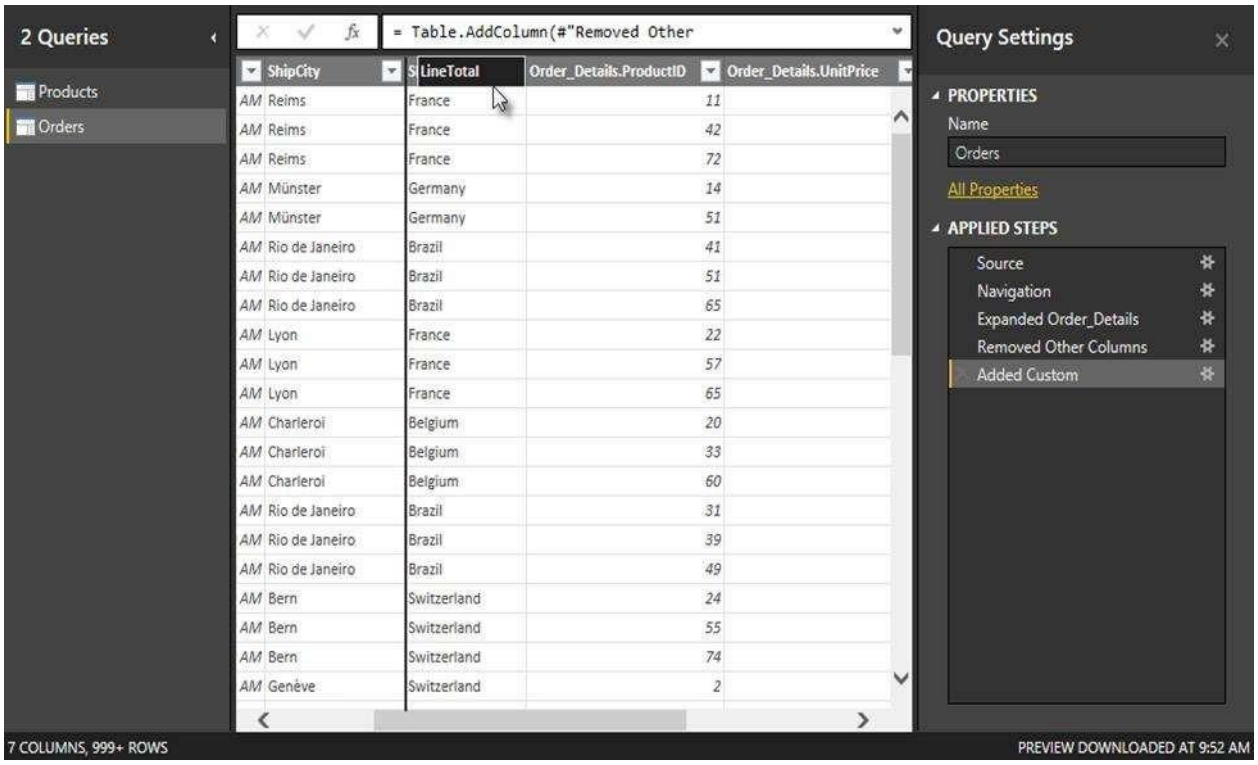


Step 6: Rename and reorder columns in the query

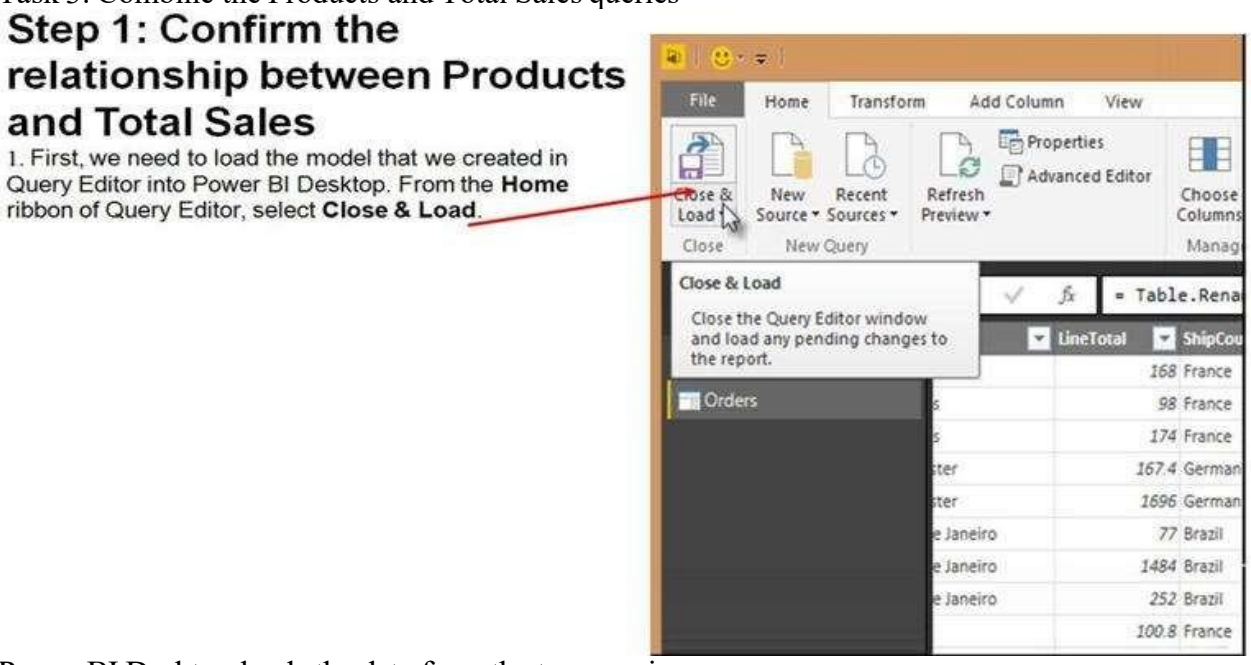
1. In Query Editor, drag the LineTotal column to the left, after ShipCountry.



2. Remove the Order_Details. prefix from the Order_Detai1s.ProductID, Order Details.UnitPrice and Order Details.Quantity columns, by double-clicking on each column header, and then deleting that text from the column name.

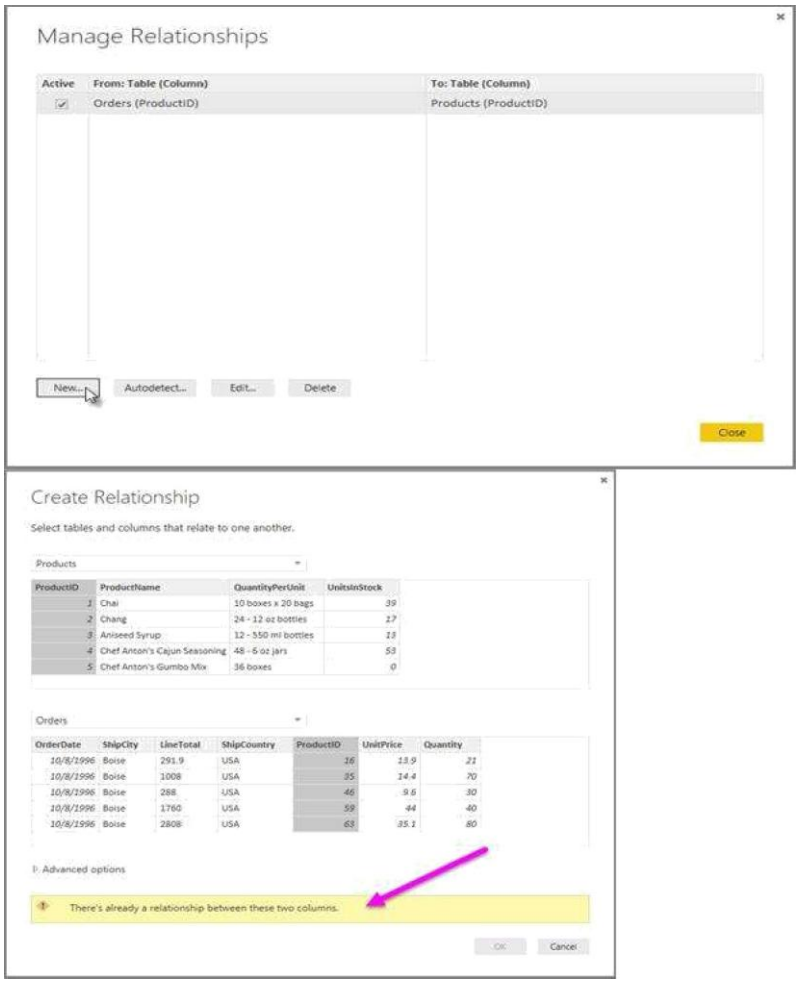


Task 3: Combine the Products and Total Sales queries

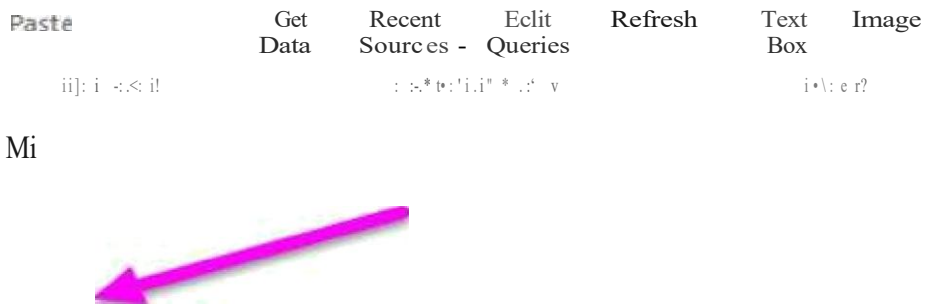


Power BI Desktop loads the data from the two queries

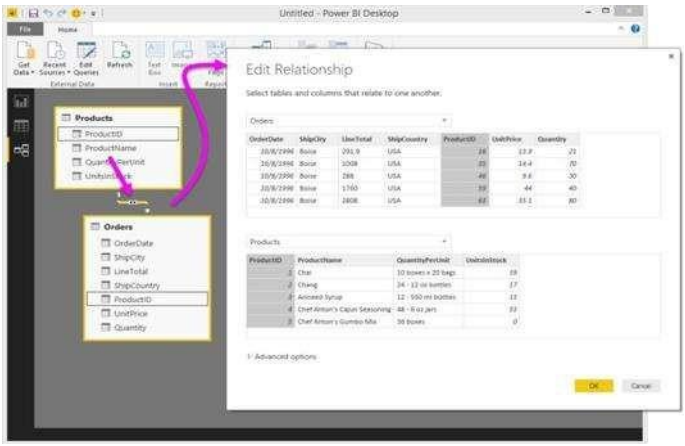
1. Once the data is loaded, select the Manage Relationships button Home ribbon
2. Select the New... button
3. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have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



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

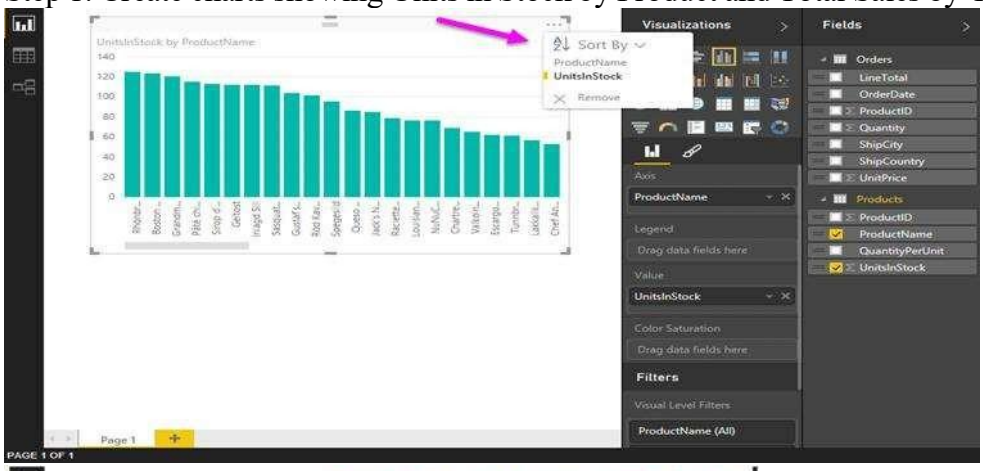


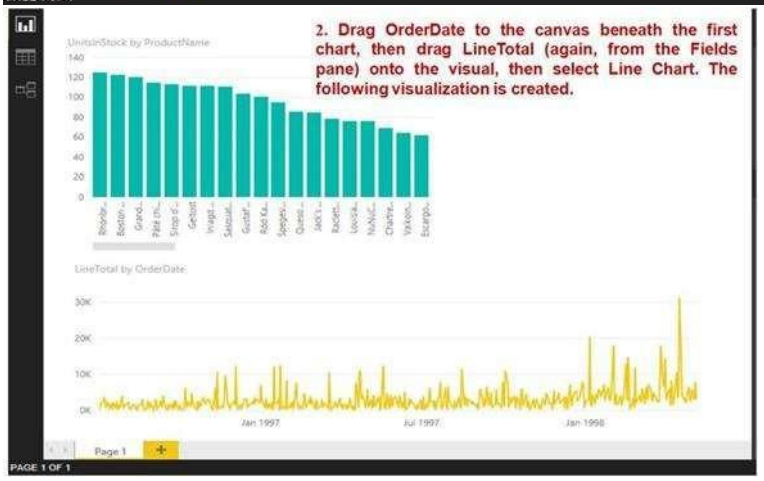
Mi



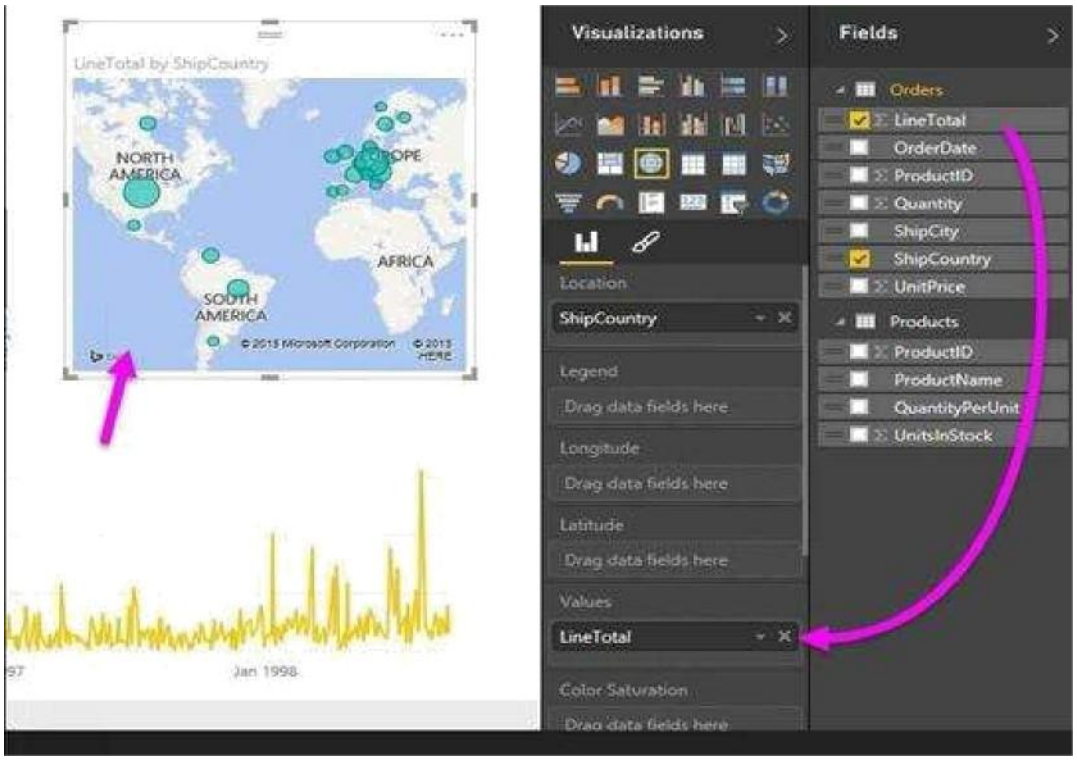
Task 4: Build visuals using your data

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





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



Step 2: Interact with your report visuals to analyze further