INDEX

Practical 1

Creating Data Model using Cassandra.

Cassandra Data Model

Step-1:

- Open a folder Datascience\apache-cassandra-3.11.4-bin\apache-cassandra-3.11.4\bin\cassandra.bat

- now open IDLE (PYTHON GUI)

- go to file -> open -> select (Datascience\apache-cassandra-3.11.4-bin\apache-cassandra-3.11.4\bin\select-cqlsh.py
-inside sqlsh.py -> run -> run module

Step-2: command to Create keyspace:

create keyspace DATASCI WITH replication={'class':'SimpleStrategy','replication_factor':3};

step – 3: command to use keyspace run this command

cqlsh> use datasci;

step – 4: command to create a new table
cqlsh:datasci> create table student(
student_id int PRIMARY KEY,
student_name text,
student_city text,
student_fees varint,
student_phone varin
Step – 5: command to display created keyspace list
Desc keyspace;
Step – 6: command to alter keyspace

alter keyspace datasci with replication={'class':'SimpleStrategy','replication_factor':2};

step – 7 : command to display all the tables of the keyspaces

cqlsh:datasci> desc tables;

step- 8: command to alter table

cqlsh:datasci>

alter table student

add student_gender text;

step- 9: command to insert data into table

insert into student(student_id,student_city,student_fees,student_name,student_phone)

values(1,'Bhy',5000,'pooja',0939293939);

(you can only add one value at a time)

Step – 10: command to show the table

cqlsh:datasci> select * from student;

step – 11: command to update table

cqlsh:datasci> update student set student_fees=200000,student_name='hima' where student_id=2;

step – 12: command to refresh the table

cqlsh:datasci> truncate student;**step – 13: command to delete the specific column data from the table** cqlsh:datasci> delete student_city from student where student_id=2;

PRACTICAL NO2

A.Text delimited CSVto HORUS format

```
import pandas as pd
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
ProcessData=InputData
ProcessData.drop(['ISO-3-Code', 'ISO-2-CODE'], axis=1,inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber', 'Country': 'CountryName'}, inplace=True)
ProcessData.set_index('CountryNumber', inplace=True)
ProcessData.sort_values('CountryName', axis=0, ascending=True, inplace=True)
print(ProcessData.head(10))
```

```
================ RESTART: C:\gaurav\pra
                    CountryName
CountryNumber
4                     Afghanistan
248                 Aland Islands
8                         Albania
12                        Algeria
16                 American Samoa
20                        Andorra
24                         Angola
660                       Anguilla
10                     Antarctica
28             Antigua and Barbuda
```

B>**XML to HORUS Format**

```
# Utility Start XML to HORUS ==================================
# Standard Tools
import pandas as pd
import xml.etree.ElementTree as ET
def df2xml(data):
    header = data.columns
    root = ET.Element('root')
    for row in range(data.shape[0]):
        entry = ET.SubElement(root,'entry')
        for index in range(data.shape[1]):
            schild=str(header[index])
            child = ET.SubElementlement(entry, schild)
            if str(data[schild][row]) != 'nan':
                child.text = str(data[schild][row])
            else:
                child.text = 'n/a'
                entry.append(child)
    result = ET.tostring(root)
    return result
def xml2df(xml_data):
    root = ET.XML(xml_data)
    all_records = []
    for i, child in enumerate(root):
                record = {}
                for subchild in child:
                    record[subchild.tag] = subchild.text
                    all_records.append(record)
    return pd.DataFrame(all_records)
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()

print('Input Data Values ===================================')
ProcessDataXML=InputData
ProcessData=xml2df(ProcessDataXML)
```

```
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
ProcessData.set_index('CountryNumber', inplace=True)
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values ================================')
print(ProcessData.head(5))
print('====================================================')
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False, encoding="latin-1")
print('XML to HORUS - Done')
```

```
============== RESTART: C:/gaurav/practicals/XML_TO_HRS_COPY.py ==============
Process Data Values ================================
              CountryName
CountryNumber
716               Zimbabwe
716               Zimbabwe
716               Zimbabwe
716               Zimbabwe
716               Zimbabwe
====================================================
XML to HORUS - Done
>>> |
```

C>JSON to HORUS Format

```
import pandas as pd
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.json'
InputData=pd.read_json(sInputFileName, orient='index', encoding="latin-1")
ProcessData=InputData
ProcessData.drop(['ISO-3-Code', 'ISO-2-CODE'], axis=1,inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber', 'Country': 'CountryName'}, inplace=True)
ProcessData.set_index('CountryNumber', inplace=True)
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print(ProcessData.head(5))
OutputData=ProcessData
sOutputFileName='c:/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False, encoding="latin-1")
print('JSON to HORUS - Done')
```

```
================ RESTART: C:\gaurav\practicals\json_to_hrs.py ================
                            CountryName
CountryNumber
716                            Zimbabwe
894                              Zambia
887                               Yemen
732                      Western Sahara
876            Wallis and Futuna Islands
JSON to HORUS - Done
>>> |
```

D>MySql Database to HORUS Format

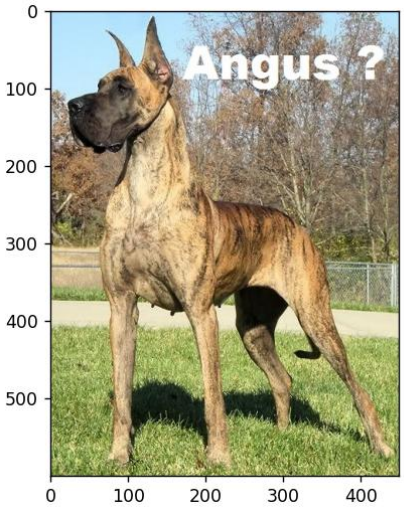
```
import pandas as pd
import sqlite3 as sq
conn = sq.connect('C:/VKHCG/05-DS/9999-Data/utility.db')
sSQL='select * FROM ' + 'Country_Code' + ';'
InputData=pd.read_sql_query(sSQL, conn)
ProcessData=InputData
ProcessData.drop(['ISO-3-Code', 'ISO-2-CODE'], axis=1,inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber', 'Country': 'CountryName'}, inplace=True)
ProcessData.set_index('CountryNumber', inplace=True)
```

```
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values ================================')
print(ProcessData.head(5))
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False, encoding="latin-1")
print('Database to HORUS - Done')
```

```
================= RESTART: C:/gaurav/practicals/db_to_hs.py =================
Process Data Values ================================
              index                CountryName
CountryNumber
716             246                   Zimbabwe
894             245                     Zambia
887             244                      Yemen
732             243             Western Sahara
876             242  Wallis and Futuna Islands
Database to HORUS - Done
>>>
```

E>Picture (JPEG) to HORUS Format

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import imageio

sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/Angus.jpg'
InputData = imageio.imread(sInputFileName, mode='RGBA')
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']
ProcessData.columns=sColumns
ProcessData.index.names =['ID']
plt.imshow(InputData)
plt.show()
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Picture to HORUS - Done')
print('=====================================================')
```




```
=====================================================
Picture to HORUS - Done
=====================================================
(myenv) PS C:\Users\GAURAV\Desktop\Resume>
```

F>Video to HORUS Format

1st ==========

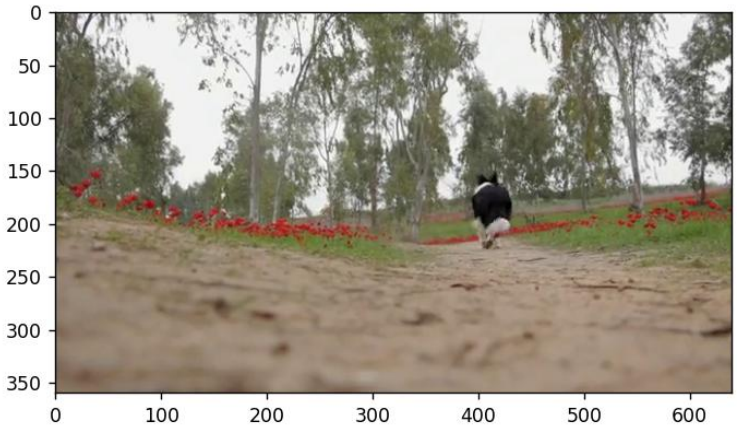
```
import os
import shutil
import cv2
sInputFileName = 'C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/dog.mp4'
sDataBaseDir = 'C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp'
if os.path.exists(sDataBaseDir):
    shutil.rmtree(sDataBaseDir)
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
vidcap = cv2.VideoCapture(sInputFileName)
if not vidcap.isOpened():
    print('Error: Could not open video file')
    exit()
count = 0
while True:
    success, image = vidcap.read()
    if not success:
        break
    sFrame = sDataBaseDir + '/dog-frame-' + str(format(count, '04d')) + '.jpg'
    print('Extracted: ', sFrame)
    cv2.imwrite(sFrame, image)
    if os.path.getsize(sFrame) == 0:
        os.remove(sFrame)
        print('Removed: ', sFrame)
        continue
    count += 1
    if cv2.waitKey(10) == 27:
        break
print('Generated: ', count, ' Frames')
print('=====================================================')
print('Movie to Frames HORUS - Done')
print('=====================================================')
```

```
(myenv) PS C                    me> python C:\Users\GAURAV\Desktop\Resume\DSPracticals\video_to_hrs.py
Extracted:   Open file in editor (ctrl + click)   ume/05-DS/05-DS/9999-Data/temp/dog-frame-0000.jpg
Extracted:   C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp/dog-frame-0001.jpg
Extracted:   C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp/dog-frame-0002.jpg
Extracted:   C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp/dog-frame-0003.jpg
Extracted:   C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp/dog-frame-0004.jpg
Extracted:   C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp/dog-frame-0005.jpg
Extracted:   C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp/dog-frame-0006.jpg
Extracted:   C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp/dog-frame-0007.jpg
```

2nd part ==========

```
import imageio
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
sDataBaseDir='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp'
f=0
for file in os.listdir(sDataBaseDir):
    if file.endswith(".jpg"):
        f += 1
sInputFileName=os.path.join(sDataBaseDir, file)
InputData = imageio.imread(sInputFileName, mode='RGBA')
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessFrameData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
ProcessFrameData['Frame']=file
plt.imshow(InputData)
plt.show()
ProcessData = []
if f == 1:
```

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

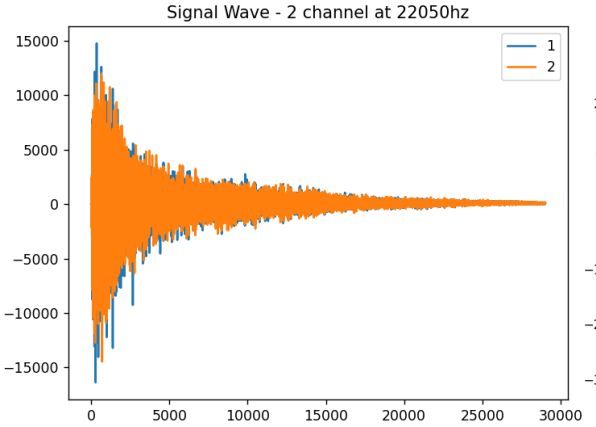


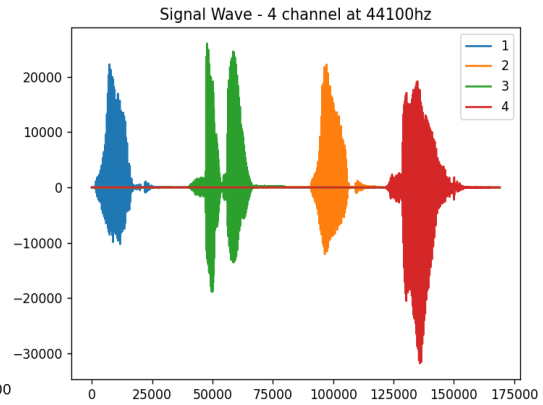
	0	1	2	3	4	5	Frame
0	94	95	89	255	183	184	dog-frame-0048.jpg
1	178	255	207	208	202	255	dog-frame-0048.jpg
2	114	115	109	255	98	99	dog-frame-0048.jpg
3	93	255	163	164	158	255	dog-frame-0048.jpg
4	155	156	150	255	175	176	dog-frame-0048.jpg
...	...	...	...	...	...	...	...
153595	108	255	151	129	106	255	dog-frame-0048.jpg
153596	150	128	107	255	149	127	dog-frame-0048.jpg
153597	106	255	149	127	106	255	dog-frame-0048.jpg
153598	150	128	107	255	151	129	dog-frame-0048.jpg
153599	108	255	151	129	108	255	dog-frame-0048.jpg

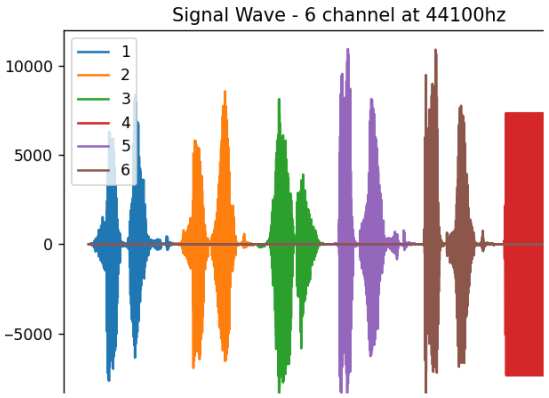
G. Audio to HORUS Format

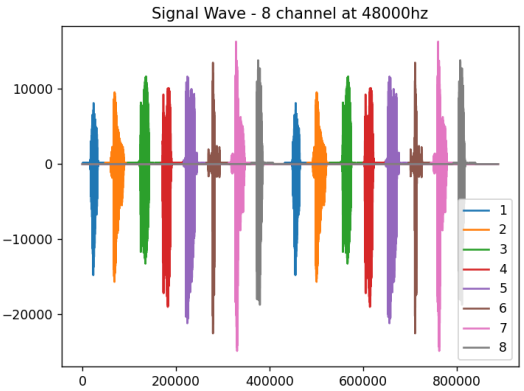
```
from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
def show_info(aname, a,r):
    print ('    ')
    print ("Audio:", aname)
    print ('    ')
    print ("Rate:", r)
    print ('    ')
    print ("shape:", a.shape)
    print ("dtype:", a.dtype)
    print ("min, max:", a.min(), a.max())
    print ('    ')
    plot_info(aname, a,r)
def plot_info(aname, a,r):
    sTitle= 'Signal Wave - '+ aname + ' at ' + str(r) + 'hz'
    plt.title(sTitle)
    sLegend=[]
    for c in range(a.shape[1]):
        sLabel = 'Ch' + str(c+1)
```

```
    sLegend=sLegend+[str(c+1)]
    plt.plot(a[:,c], label=sLabel)
  plt.legend(sLegend)
  plt.show()
#==============================================================
sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/2ch-sound.wav'
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/4ch-sound.wav'
InputRate, InputData = wavfile.read(sInputFileName)
show_info("4 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/6ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("6 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
 #==============================================================
sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/8ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("8 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Audio to HORUS - Done')
```










```
==================================================
Processing :  C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/8ch-sound.wav
==================================================

Audio: 8 channel

Rate: 48000

shape: (888000, 8)
dtype: int16
min, max: -24859 16303
```

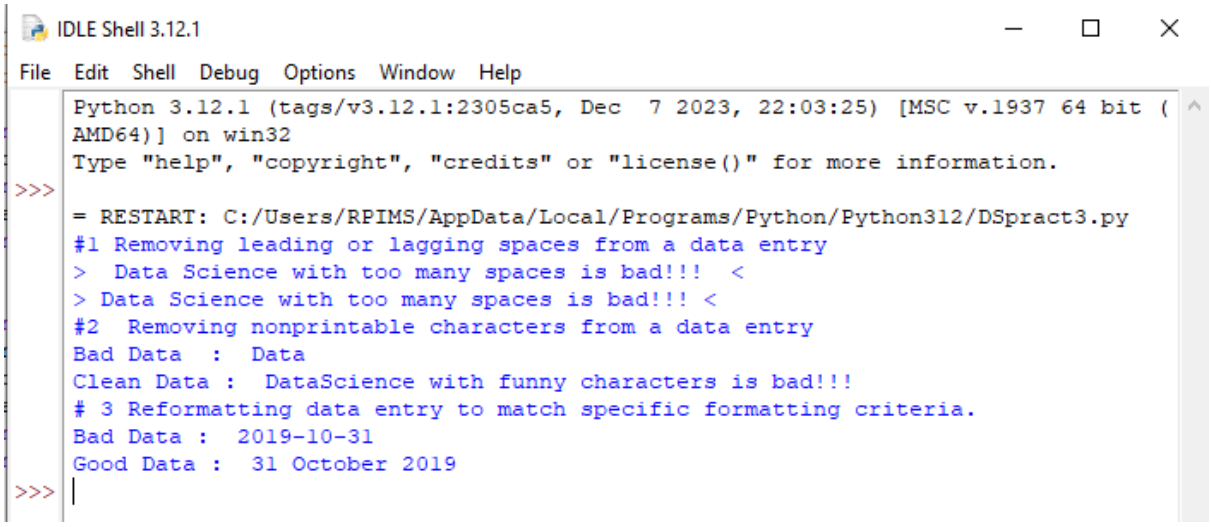
Practical 3: Utilities and Auditing

Basic Utility Design

A. Fixers Utilities:

Fixers enable your solution to take your existing data and fix a specific quality issue.

```
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print('#1 Removing leading or lagging spaces from a data entry');
 baddata = " Data Science with too many spaces is bad!!! " print('>',baddata,'<')
cleandata=baddata.strip() print('>',cleandata,'<')

# 2 Removing nonprintable characters from a data entry
print('#2  Removing nonprintable characters from a data entry')
printable = set(string.printable)
baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!" cleandata=".join(filter(lambda x:
x in string.printable,baddata))
print('Bad Data  : ',baddata);
print('Clean Data : ',cleandata)

# 3 Reformatting data entry to match specific formatting criteria.
#  Convert YYYY/MM/DD to DD Month YYYY
print('# 3 Reformatting data entry to match specific formatting criteria.')
 baddate = dt.date(2019, 10, 31)
baddata=format(baddate,'%Y-%m-%d')
gooddate = dt.datetime.strptime(baddata,'%Y-%m-%d')
gooddata=format(gooddate,'%d %B %Y')
print('Bad Data : ',baddata)
print('Good Data : ',gooddata)
```

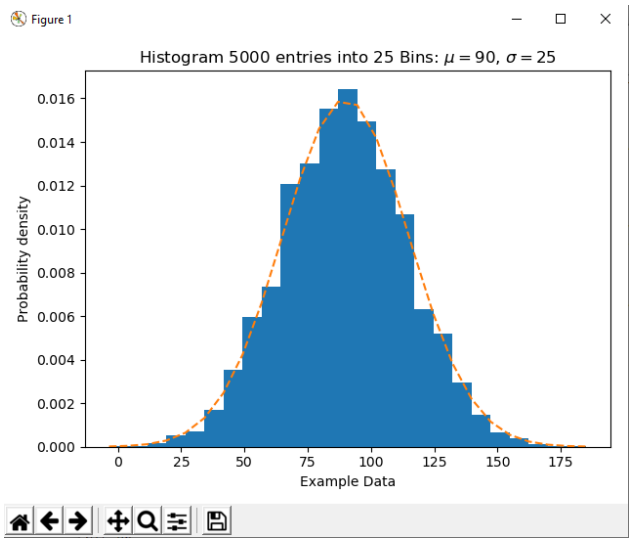


B. Data Binning or Bucketing

```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
np.random.seed(0)
# example data
mu = 90  # mean of distribution
```

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

Output:



C. Averaging of Data

Input:

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

Output:

```
======================= RESTART: C:/Users/RPIMS/AppData/Loca
Working Base : C:/VKHCG  using
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
     Country Place_Name  Latitude
0         US   New York   40.7528
1         US   New York   40.7528
2         US   New York   40.7528
3         US   New York   40.7528
4         US   New York   40.7528
...      ...        ...       ...
3557      DE     Munich   48.0915
3558      DE     Munich   48.1833
3559      DE     Munich   48.1000
3560      DE     Munich   48.1480
3561      DE     Munich   48.1480

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

D. Outlier Detection

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

Output:

```
= RESTART: C:/Users/RPIMS/AppData/Local/Programs/Python/Pyth
Working Base : C:/VKHCG
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
All Data
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
...      ...        ...       ...
3434      GB     London   51.5092
3435      GB     London   51.5092
3436      GB     London   51.5163
3437      GB     London   51.5085
3438      GB     London   51.5136

[1502 rows x 3 columns]
Outliers

Warning (from warnings module):
  File "C:/Users/RPIMS/AppData/Local/Programs/Python/Python3
line 17
    UpperBound=float(MeanData+StdData)
FutureWarning: Calling float on a single element Series is de
aise a TypeError in the future. Use float(ser.iloc[0]) instea
Higher than  51.512635507867415
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
1916      GB     London   51.5491
1919      GB     London   51.5161
1920      GB     London   51.5198
```

```
1921      GB     London   51.5198
1923      GB     London   51.5237
1924      GB     London   51.5237
1925      GB     London   51.5237
1926      GB     London   51.5237
1927      GB     London   51.5232
3436      GB     London   51.5163
3438      GB     London   51.5136

Warning (from warnings module):
  File "C:/Users/RPIMS/AppData/Loca
line 21
    LowerBound=float(MeanData-StdDa
FutureWarning: Calling float on a s
aise a TypeError in the future. Use
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]
```

Audit

E. Logging

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

for sCompany in sCompanies:
  sFileDir=Base + '/' + sCompany
  if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
  for sLayer in sLayers:
    log = logging.getLogger()
    for hdlr in log.handlers[:]:
      log.removeHandler(hdlr)
    sFileDir=Base + '/' + sCompany + '/' + sLayer + '/Logging'
    if os.path.exists(sFileDir):
      shutil.rmtree(sFileDir)
    time.sleep(2)
```

```
    if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
    skey=str(uuid.uuid4())
    sLogFile=Base + '/' + sCompany + '/' + sLayer + '/Logging/Logging_'+skey+'.log'
    print('Set up:',sLogFile)
    logging.basicConfig(level=logging.DEBUG,
                        format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
                        datefmt='%m-%d %H:%M',
                        filename=sLogFile,
                        filemode='w')
    console = logging.StreamHandler()
    console.setLevel(logging.INFO)
    formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
    console.setFormatter(formatter)
    logging.getLogger('').addHandler(console)
    logging.info('Practical Data Science is fun!.')
    for sLevel in sLevels:
        sApp='Apllication-'+ sCompany + '-' + sLayer + '-' + sLevel
        logger = logging.getLogger(sApp)
        if sLevel == 'debug':
            logger.debug('Practical Data Science logged a debugging message.')
        if sLevel == 'info':
            logger.info('Practical Data Science logged information message.')
        if sLevel == 'warning':
            logger.warning('Practical Data Science logged a warning message.')
        if sLevel == 'error':
            logger.error('Practical Data Science logged an error message.')
```

Output:

```
 RESTART: D:\yukta\Datascience\Datascience\VKHCG\practical-data-science\VKHCG\77
-Yoke\Yoke_Logging.py
('Set up:', 'C:/VKHCG/01-Vermeulen/01-Retrieve/Logging/Logging_e48e608b-23f3-43e
d-885d-5eff531f83ad.log')
root         : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-01-Retrieve-info: INFO     Practical Data Science logge
d information message.
Apllication-01-Vermeulen-01-Retrieve-warning: WARNING  Practical Data Science lo
gged a warning message.
Apllication-01-Vermeulen-01-Retrieve-error: ERROR    Practical Data Science logg
ed an error message.
```

Practical 4

A.Perform the following data processing using R.

Code

```
library(readr)
IP_DATA_ALL <- read_csv("E:/NIKHILESH/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
View(IP_DATA_ALL)
```

Output

```
spec(IP_DATA_ALL)

set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)

IP_DATA_ALL_FIX <- read.csv("E:/NIKHILESH/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/01-R/IP_DATA_ALL_FIX.csv")
sapply(IP_DATA_ALL_FIX, typeof)

library(data.table)
hist_country=data.table(Country=unique(IP_DATA_ALL_FIX[is.na(IP_DATA_ALL_FIX ['Country']) == 0, ]$Country))
setorder(hist_country,'Country')
hist_country_with_id=rowid_to_column(hist_country, var = "RowIDCountry")
View(hist_country_fix)
IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_FIX, table(Country)))
View(IP_DATA_COUNTRY_FREQ)

sapply(IP_DATA_ALL_FIX[,'Latitude'], min, na.rm=TRUE)

sapply(IP_DATA_ALL_FIX[,'Country'], min, na.rm=TRUE)

sapply(IP_DATA_ALL_FIX[,'Latitude'], max, na.rm=TRUE)
```

Finding mean median range and quantile following are the commands are used-

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

Finding the standard deviation of any column in table the commands will be –

```
sapply(IP_DATA_ALL_FIX [,'Latitude'], sd, na.rm=TRUE)
```

B. Program to retrieve different attributes of data.

Code-

```
import sys
import os
import pandas as pd

sFileName='E:/NIKHILESH/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
sFileDir='E:/NIKHILESH/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
print('Rows:', IP_DATA_ALL.shape[0])
print('Columns:', IP_DATA_ALL.shape[1])
print('### Raw Data Set ###')
for i in range(0,len(IP_DATA_ALL.columns)):
 print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
```

```
print('### Fixed Data Set ###')
IP_DATA_ALL_FIX=IP_DATA_ALL
for i in range(0,len(IP_DATA_ALL.columns)):
 cNameOld=IP_DATA_ALL_FIX.columns[i] + ' '
 cNameNew=cNameOld.strip().replace(" ", ".")
IP_DATA_ALL_FIX.columns.values[i] = cNameNew
print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print('Fixed Data Set with ID')
IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
IP_DATA_ALL_with_ID.index.names = ['RowID']
sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")
print('### Done!! ###')
```

Output-

C. Data Pattern

Code

Write the program using r Studio

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

output

D. Loading IP_DATA_ALL:

Code

```
import sys
import os
import pandas as pd
Base='C:/VKHCG'
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
print('Rows:', IP_DATA_ALL.shape[0])
```

```
print('Columns:', IP_DATA_ALL.shape[1])
print('### Raw Data Set ####################################')
for i in range(0,len(IP_DATA_ALL.columns)):
print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print('### Fixed Data Set ##################################')
IP_DATA_ALL_FIX=IP_DATA_ALL
for i in range(0,len(IP_DATA_ALL.columns)):
cNameOld=IP_DATA_ALL_FIX.columns[i] + ' '
cNameNew=cNameOld.strip().replace(" ", ".")
IP_DATA_ALL_FIX.columns.values[i] = cNameNew
print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
#print(IP_DATA_ALL_FIX.head())
print('Fixed Data Set with ID')
IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
IP_DATA_ALL_with_ID.index.names = ['RowID']
#print(IP_DATA_ALL_with_ID.head())
sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")
print('### Done!! ##############################################')
output
```

Vermeulen PLC

Code

```
import sys
import os
import pandas as pd
from math import radians, cos, sin, asin, sqrt
# Function to calculate haversine distance
def haversine(lon1, lat1, lon2, lat2, stype):
    # Convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * asin(sqrt(a))
    # Determine the radius of Earth based on the unit type
    if stype == 'km':
        r = 6371  # Radius of Earth in kilometers
    else:
        r = 3956  # Radius of Earth in miles

    # Calculate and return the distance
    d = round(c * r, 3)
    return d
# File paths
sFileName = 'E:/NIKHILESH/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv'
sFileDir = 'E:/NIKHILESH/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
# Check if output directory exists; create if not
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
# Load the CSV file
print('Loading:', sFileName)
IP_DATA_ALL = pd.read_csv(
    sFileName,
    header=0,
    low_memory=False,
```

```
    usecols=['Country', 'Place Name', 'Latitude', 'Longitude'],
    encoding="latin-1"
)
# Process the data
IP_DATA = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
IP_DATA.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA1 = IP_DATA.copy()
IP_DATA1.insert(0, 'K', 1)
IP_DATA2 = IP_DATA1.copy()
# Cross-join to calculate pairwise distances
IP_CROSS = pd.merge(right=IP_DATA1, left=IP_DATA2, on='K')
IP_CROSS.drop('K', axis=1, inplace=True)
# Rename columns for clarity
IP_CROSS.rename(columns={
    'Longitude_x': 'Longitude_from', 'Longitude_y': 'Longitude_to',
    'Latitude_x': 'Latitude_from', 'Latitude_y': 'Latitude_to',
    'Place_Name_x': 'Place_Name_from', 'Place_Name_y': 'Place_Name_to',
    'Country_x': 'Country_from', 'Country_y': 'Country_to'
}, inplace=True)
# Calculate distances in kilometers and miles
IP_CROSS['DistanceBetweenKilometers'] = IP_CROSS.apply(
    lambda row: haversine(
        row['Longitude_from'],
        row['Latitude_from'],
        row['Longitude_to'],
        row['Latitude_to'],
        'km'
    ),
    axis=1
)

IP_CROSS['DistanceBetweenMiles'] = IP_CROSS.apply(
    lambda row: haversine(
        row['Longitude_from'],
        row['Latitude_from'],
        row['Longitude_to'],
        row['Latitude_to'],
        'miles'
    ),
    axis=1
)
# Save the result to a CSV file
print('Saving results...')
sFileName2 = os.path.join(sFileDir, 'Retrieve_IP_Routing.csv')
IP_CROSS.to_csv(sFileName2, index=False, encoding="latin-1")
print('### Done!! ##############################################')
```

output –

See the file named Retrieve_IP_Routing.csv in C:\VKHCG\01-Vermeulen\01-Retrieve\01-EDS\02-
Total Records: 22501

So, the distance between a router in New York (40.7528, -73.9725) to anoher router in New York (40.7214, -74.0052) is 4.448 kilometers, or 2.762 miles.

Building a Diagram for the Scheduling of Jobs

Code

```
import sys
import os
```

```
import pandas as pd
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
sFileName='E:/NIKHILESH/VKHCG/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
sFileDir='E:/NIKHILESH/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
  os.makedirs(sFileDir)
ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])
sFileName2=sFileDir + '/' + OutputFileName
ROUTERLOC.to_csv(sFileName2, index = False, encoding="latin-1")
print('### Done!! ############################################')
```

output

Understanding Your Online Visitor Data

Code

```
import sys
import os
import pandas as pd
import gzip as gz

InputFileName='IP_DATA_ALL.csv'
OutputFileName='Retrieve_Online_Visitor'
CompanyIn= '01-Vermeulen'
CompanyOut= '02-Krennwallner'
Base='E:/NIKHILESH/VKHCG/'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
Base='E:/NIKHILESH/VKHCG/'
sFileName=Base + '/' + CompanyIn + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,usecols=['Country','Place.Name','Latitude','Longitude','Fi
rst.IP.Number','Last.IP.Number'])
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA_ALL.rename(columns={'First IP Number': 'First_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Last IP Number': 'Last_IP_Number'}, inplace=True)
sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
  os.makedirs(sFileDir)
visitordata = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
visitordata10=visitordata.head(10)
print('Rows :',visitordata.shape[0])
print('Columns :',visitordata.shape[1])
print('Export CSV')
sFileName2=sFileDir + '/' + OutputFileName + '.csv'
visitordata.to_csv(sFileName2, index = False)
print('Store All:',sFileName2)
sFileName3=sFileDir + '/' + OutputFileName + '_10.csv'
visitordata10.to_csv(sFileName3, index = False)
```

```
print('Store 10:',sFileName3)
for z in ['gzip', 'bz2', 'xz']:
  if z == 'gzip':
   sFileName4=sFileName2 + '.gz'
else:
   sFileName4=sFileName2 + '.' + z
visitordata.to_csv(sFileName4, index = False, compression=z)
print('Store :',sFileName4)
print('Export JSON')
for sOrient in ['split','records','index', 'columns','values','table']:
   sFileName2=sFileDir + '/' + OutputFileName + '_' + sOrient + '.json'
visitordata.to_json(sFileName2,orient=sOrient,force_ascii=True)
print('Store All:',sFileName2)
sFileName3=sFileDir + '/' + OutputFileName + '_10_' + sOrient + '.json'
visitordata10.to_json(sFileName3,orient=sOrient,force_ascii=True)
print('Store 10:',sFileName3)
sFileName4=sFileName2 + '.gz'
file_in = open(sFileName2, 'rb')
file_out = gz.open(sFileName4, 'wb')
file_out.writelines(file_in)
file_in.close()
file_out.close()
print('Store GZIP All:',sFileName4)
sFileName5=sFileDir + '/' + OutputFileName + '_' + sOrient + '_UnGZip.json'
file_in = gz.open(sFileName4, 'rb')
file_out = open(sFileName5, 'wb')
file_out.writelines(file_in)
file_in.close()
file_out.close()
print('Store UnGZIP All:',sFileName5)
print('### Done!! ##############################################')
output
```

XML processing

Code

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

```
    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)
InputFileName='IP_DATA_ALL.csv'
OutputFileName='Retrieve_Online_Visitor.xml'
CompanyIn= '01-Vermeulen'
CompanyOut= '02-Krennwallner'
if sys.platform == 'linux':
  Base=os.path.expanduser('~') + '/VKHCG'
else:
  Base='E:/NIKHILESH/VKHCG/'

print('Working Base :',Base, ' using ', sys.platform)
sFileName=Base + '/' + CompanyIn + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False)
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA_ALL.rename(columns={'First IP Number': 'First_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Last IP Number': 'Last_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Post Code': 'Post_Code'}, inplace=True)
sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
  os.makedirs(sFileDir)
visitordata = IP_DATA_ALL.head(10000)
print('Original Subset Data Frame')
print('Rows :',visitordata.shape[0])
print('Columns :',visitordata.shape[1])
print(visitordata)
print('Export XML')
sXML=df2xml(visitordata)
sFileName=sFileDir + '/' + OutputFileName
file_out = open(sFileName, 'wb')
file_out.write(sXML)
file_out.close()
print('Store XML:',sFileName)
xml_data = open(sFileName).read()
unxmlrawdata=xml2df(xml_data)
print('Raw XML Data Frame')
print('Rows :',unxmlrawdata.shape[0])
print('Columns :',unxmlrawdata.shape[1])
print(unxmlrawdata)
unxmldata = unxmlrawdata.drop_duplicates(subset=None, keep='first', inplace=False)
print('Deduplicated XML Data Frame')
print('Rows :',unxmldata.shape[0])
print('Columns :',unxmldata.shape[1])
print(unxmldata)
#print('### Done!!#########')
```

Output

Adopt New Shipping Containers

Code

```
import sys
import os
import pandas as pd
ContainerFileName = 'Retrieve_Container.csv'
BoxFileName = 'Retrieve_Box.csv'
ProductFileName = 'Retrieve_Product.csv'
Company = '03-Hillman'
Base = 'E:/NIKHILESH/10th .pdfVKHCG'
print('Working Base :', Base, ' using ', sys.platform)
sFileDir = Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
containerLength = range(1, 21)
containerWidth = range(1, 10)
containerHeigth = range(1, 6)
containerStep = 1
c = 0
# Initialize an empty DataFrame for containers
ContainerFrame = pd.DataFrame()
for l in containerLength:
    for w in containerWidth:
        for h in containerHeigth:
            containerVolume = (l / containerStep) * (w / containerStep) * (h / containerStep)
            c += 1
            ContainerLine = {
                'ShipType': 'Container',
                'UnitNumber': 'C' + format(c, "06d"),
                'Length': round(l, 4),
                'Width': round(w, 4),
                'Height': round(h, 4),
                'ContainerVolume': round(containerVolume, 6)
            }
            ContainerRow = pd.DataFrame([ContainerLine])
            ContainerFrame = pd.concat([ContainerFrame, ContainerRow], ignore_index=True)

ContainerFrame.index.name = 'IDNumber'
print('################')
print('## Container')
print('################')
print('Rows :', ContainerFrame.shape[0])
print('Columns :', ContainerFrame.shape[1])

sFileContainerName = sFileDir + '/' + ContainerFileName
ContainerFrame.to_csv(sFileContainerName, index=False)
boxLength = range(1, 21)
boxWidth = range(1, 21)
boxHeigth = range(1, 21)
packThick = range(0, 6)
boxStep = 10
b = 0
# Initialize an empty DataFrame for boxes
BoxFrame = pd.DataFrame()
for l in boxLength:
    for w in boxWidth:
        for h in boxHeigth:
            for t in packThick:
```

```
            boxVolume = round((l / boxStep) * (w / boxStep) * (h / boxStep), 6)
            productVolume = round(((l - t) / boxStep) * ((w - t) / boxStep) * ((h - t) / boxStep), 6)
            if productVolume > 0:
                b += 1
                BoxLine = {
                    'ShipType': 'Box',
                    'UnitNumber': 'B' + format(b, "06d"),
                    'Length': round(l / 10, 6),
                    'Width': round(w / 10, 6),
                    'Height': round(h / 10, 6),
                    'Thickness': round(t / 5, 6),
                    'BoxVolume': round(boxVolume, 9),
                    'ProductVolume': round(productVolume, 9)
                }
                BoxRow = pd.DataFrame([BoxLine])
                BoxFrame = pd.concat([BoxFrame, BoxRow], ignore_index=True)

BoxFrame.index.name = 'IDNumber'
print('## Box####')
print('Rows :', BoxFrame.shape[0])
print('Columns :', BoxFrame.shape[1])

sFileBoxName = sFileDir + '/' + BoxFileName
BoxFrame.to_csv(sFileBoxName, index=False)
productLength = range(1, 21)
productWidth = range(1, 21)
productHeigth = range(1, 21)
productStep = 10
p = 0
# Initialize an empty DataFrame for products
ProductFrame = pd.DataFrame()
for l in productLength:
    for w in productWidth:
        for h in productHeigth:
            productVolume = round((l / productStep) * (w / productStep) * (h / productStep), 6)
            if productVolume > 0:
                p += 1
                ProductLine = {
                    'ShipType': 'Product',
                    'UnitNumber': 'P' + format(p, "06d"),
                    'Length': round(l / 10, 6),
                    'Width': round(w / 10, 6),
                    'Height': round(h / 10, 6),
                    'ProductVolume': round(productVolume, 9)
                }
                ProductRow = pd.DataFrame([ProductLine])
                ProductFrame = pd.concat([ProductFrame, ProductRow], ignore_index=True)

ProductFrame.index.name = 'IDNumber'
print('## Product')
print('Rows :', ProductFrame.shape[0])
print('Columns :', ProductFrame.shape[1])

sFileProductName = sFileDir + '/' + ProductFileName
ProductFrame.to_csv(sFileProductName, index=False)
```

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

Global Post Codes

Code in r studio

```
library(readr)
All_Countries <- read_delim("C:/VKHCG/03-Hillman/00-RawData/All_Countries.txt",
"\t", col_names = FALSE,
col_types = cols(
X12 = col_skip(),
X6 = col_skip(),
X7 = col_skip(),
X8 = col_skip(),
X9 = col_skip()),
na = "null", trim_ws = TRUE)
write.csv(All_Countries,
file = "C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_All_Countries.csv")
output
```

The program will successfully uploaded a new file named Retrieve_All_Countries.csv, after removing column No. 6, 7, 8, 9 and 12 from All_Countries.txt

Program to connect to different data sources.

Code

```
import sqlite3 as sq
import pandas as pd
Base='C:/VKHCG'
sDatabaseName=Base + '/01-Vermeulen/00-RawData/SQLite/vermeulen.db'
conn = sq.connect(sDatabaseName)
sFileName='C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
print('Loading :',sFileName)
IP_DATA_ALL_FIX=pd.read_csv(sFileName,header=0,low_memory=False)
IP_DATA_ALL_FIX.index.names = ['RowIDCSV']
sTable='IP_DATA_ALL'
print('Storing :',sDatabaseName,' Table:',sTable)
IP_DATA_ALL_FIX.to_sql(sTable, conn, if_exists="replace")
print('Loading :',sDatabaseName,' Table:',sTable)
TestData=pd.read_sql_query("select * from IP_DATA_ALL;", conn)
print('## Data Values')
print(TestData)\
```

Practical no 5

Assessing Data

Perform error management on the given data using pandas package.

Drop the Columns Where All Elements Are Missing Values

Code:-

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

print('Working Base :',Base, ' using ', sys.platform)
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-01.csv'
```

```
Company='01-Vermeulen'
Base='C:/VKHCG'
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('## Raw Data Values')
print(RawData)
print('## Data Profile')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
TestData=RawData.dropna(axis=1, how='all')
print('## Test Data Values')
print(TestData)
print('## Data Profile')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
print('### Done!! ######################')
```

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

Code:-

```
import sys
import os
import pandas as pd
pd.options.mode.chained_assignment = None
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sInputFileName3='01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv'
sOutputFileName='Assess-Network-Routing-Company.csv'
Company='01-Vermeulen'
### Import Country Data
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('Loading :',sFileName)
CountryData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
## Assess Country Data
print('Changed :',CountryData.columns.values)
CountryData.rename(columns={'Country': 'Country_Name'}, inplace=True)
CountryData.rename(columns={'ISO-2-CODE': 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inplace=True)
CountryData.drop('ISO-3-Code', axis=1, inplace=True)
CountryData.drop('RowID', axis=1, inplace=True)
print('To :',CountryData.columns.values)
### Import Company Data
sFileName=Base + '/' + Company + '/' + sInputFileName2
```

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

Code:-

```
####################Assess-Network-Routing- Customer.py #######################
import sys
import os
import pandas as pd
pd.options.mode.chained_assignment = None
Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
sInputFileName=Base+'/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-
Customer.csv'
sOutputFileName='Assess-Network-Routing-Customer.gml'
Company='01-Vermeulen'
### Import Country Data
sFileName=sInputFileName
print('Loading :',sFileName)
CustomerData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CustomerData.columns.values)
print(CustomerData.head())
print('### Done!! #####################')
```

Output:-

Code:-

Assess-Network-Routing-Node.py

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

Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
sOutputFileName='Assess-Network-Routing-Node.csv'
Company='01-Vermeulen'
### Import IP Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('Loading :',sFileName)
IPData=pd.read_csv(sFileName,header=0,low_memory=False, 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'}, inplace=True)
IPData.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
IPData.rename(columns={'Post.Code': 'Post_Code'}, inplace=True)
IPData.rename(columns={'First.IP.Number': 'First_IP_Number'}, inplace=True)
IPData.rename(columns={'Last.IP.Number': 'Last_IP_Number'}, inplace=True)
print('To :',IPData.columns.values)
print('Change ',IPData.columns.values)
for i in IPData.columns.values:
j='Node_'+i
IPData.rename(columns={i: j}, inplace=True)
print('To ', IPData.columns.values)
```

```
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
sFileName=sFileDir + '/' + sOutputFileName
print('Storing :', sFileName)
IPData.to_csv(sFileName, index = False, encoding="latin-1")
print('### Done!! ######################')
```

Output:-

Write a Python / R program to build directed acyclic graph

Code:-

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

Code:-

```
import sys
import os
import sqlite3 as sq
import pandas as pd
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
sOutputFileName='Assess-DE-Billboard-Visitor.csv'
Company='02-Krennwallner'
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/krennwallner.db'
conn = sq.connect(sDatabaseName)
### Import Billboard Data
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('Loading :',sFileName)
BillboardRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
BillboardRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BillboardData=BillboardRawData
print('Loaded Company :',BillboardData.columns.values)
sTable='Assess_BillboardData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardData.to_sql(sTable, conn, if_exists="replace")
print(BillboardData.head())
print('Rows : ',BillboardData.shape[0])
### Import Billboard Data
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('Loading :',sFileName)
VisitorRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData[VisitorRawData.Country=='DE']
print('Loaded Company :',VisitorData.columns.values)
sTable='Assess_VisitorData'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print(VisitorData.head())
print('Rows : ',VisitorData.shape[0])
sTable='Assess_BillboardVisitorData'
```

```
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.Country AS BillboardCountry,"
sSQL=sSQL+ " A.Place_Name AS BillboardPlaceName,"
sSQL=sSQL+ " A.Latitude AS BillboardLatitude, "
sSQL=sSQL+ " A.Longitude AS BillboardLongitude,"
sSQL=sSQL+ " B.Country AS VisitorCountry,"
sSQL=sSQL+ " B.Place_Name AS VisitorPlaceName,"
sSQL=sSQL+ " B.Latitude AS VisitorLatitude, "
sSQL=sSQL+ " B.Longitude AS VisitorLongitude,"
sSQL=sSQL+ " (B.Last_IP_Number - B.First_IP_Number) * 365.25 * 24 * 12 AS VisitorYearRate"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_BillboardData as A"
sSQL=sSQL+ " JOIN "
sSQL=sSQL+ " Assess_VisitorData as B"
sSQL=sSQL+ " ON "
sSQL=sSQL+ " A.Country = B.Country"
sSQL=sSQL+ " AND "
sSQL=sSQL+ " A.Place_Name = B.Place_Name;"
BillboardVistorsData=pd.read_sql_query(sSQL, conn)
sTable='Assess_BillboardVistorsData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardVistorsData.to_sql(sTable, conn, if_exists="replace")
print(BillboardVistorsData.head())
print('Rows : ',BillboardVistorsData.shape[0])
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
print('Storing :', sFileName)
sFileName=sFileDir + '/' + sOutputFileNameBillboardVistorsData.to_csv(sFileName, index = False
```

Practical 5 Assessing Data

```
import pandas as pd

def assess_data_quality(df):
    """
    Function to assess data quality of a pandas DataFrame.
    Outputs summary statistics, missing values, duplicates, and data types.
    """

    print("\n--- Data Summary ---")
    print(df.describe(include='all'))

    print("\n--- Missing Values ---")
    missing_values = df.isnull().sum()
    print(missing_values[missing_values > 0])

    print("\n--- Duplicates ---")
    duplicate_count = df.duplicated().sum()
    print(f"Number of duplicate rows: {duplicate_count}")

    print("\n--- Data Types ---")
    print(df.dtypes)

    print("\n--- Unique Values per Column ---")
    for col in df.columns:
        print(f"{col}: {df[col].nunique()} unique values")

    print("\n--- Potential Inconsistencies (Categorical Columns) ---")
    for col in df.select_dtypes(include='object').columns:
        print(f"\nColumn '{col}' unique values:")
        print(df[col].value_counts())
# Example usage:
# Load sample data (replace with your dataset)
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie', 'Alice'],
    'Age': [25, 30, None, 25],
    'City': ['New York', 'Los Angeles', 'New York', 'New York'],
    'Salary': [70000, 80000, 75000, 70000]
})

assess_data_quality(df)
```

Output:

	Name	Age	City	Salary
count	4	3.000000	4	4.000000
unique	3	NaN	2	NaN
top	Alice	NaN	New York	NaN
freq	2	NaN	3	NaN
mean	NaN	26.666667	NaN	73750.000000
std	NaN	2.886751	NaN	4787.135539
min	NaN	25.000000	NaN	70000.000000
25%	NaN	25.000000	NaN	70000.000000
50%	NaN	25.000000	[illegible]	72500.000000
75%	NaN	27.500000	NaN	76250.000000

Practical 6 Build the time hub ,links , and satelites.

#code

```
import datetime
import time
import threading

class TimeHub:
    def __init__(self):
        self.time_links = []
    def add_time_link(self, link):
        self.time_links.append(link)
    def distribute_time(self):
        current_time = datetime.datetime.utcnow()
        for link in self.time_links:
            link.receive_time(current_time)
class TimeLink:
    def __init__(self, satellite):
        self.satellite = satellite
    def receive_time(self, current_time):
        self.satellite.update_time(current_time)
class Satellite:
    def __init__(self, name):
        self.name = name
        self.current_time = None
    def update_time(self, current_time):
        self.current_time = current_time
    def get_time(self):
        return self.current_time
def main():
    time_hub = TimeHub()
    satellite1 = Satellite("Satellite_1")
    satellite2 = Satellite("Satellite_2")
    link1 = TimeLink(satellite1)
    link2 = TimeLink(satellite2)
    time_hub.add_time_link(link1)
    time_hub.add_time_link(link2)
    def sync_time():
        while True:
            time_hub.distribute_time()
            time.sleep(1)
    sync_thread = threading.Thread(target=sync_time)
    sync_thread.daemon = True
    sync_thread.start()

    time.sleep(5)
    print(f"{satellite1.name} Time: {satellite1.get_time()}")
    print(f"{satellite2.name} Time: {satellite2.get_time()}")

if __name__ == "__main__":
```

main()

output

```
========== RESTART: C:\VKHCG\04-Clark\03-Process\Process-People.py ==========
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
################################
C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
################################
################
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process_Person ################
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Person-Gender
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Person-Names
################################
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Satellite-Person-Names.csv
################################
################################
################
Vacuum Databases
################
### Done!! ############################################
```

Practical 7 - Transform-Gunnarsson_is_Born.py

```
import sys import os
from datetime import datetime from pytz
import timezone import pandas as pd
import sqlite3 as sq import uuid
pd.options.mode.chained_assignment = None if
sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
Company='01-Vermeulen' InputDir='00-
RawData' InputFileName='VehicleData.csv'
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite' if not
os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/Vermeulen.db' conn1 =
sq.connect(sDatabaseName)
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
   os.makedirs(sDataVaultDir)
sDatabaseName=sDataVaultDir + '/datavault.db' conn2 =
sq.connect(sDatabaseName)
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
   os.makedirs(sDataWarehouseDir)
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn3 =
sq.connect(sDatabaseName)
print('Time Category') print('UTC Time')
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
print(BirthDateZoneUTCStr)
print('Birth Date in Reykjavik :')
BirthZone = 'Atlantic/Reykjavik'
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
print(BirthDateStr)
IDZoneNumber=str(uuid.uuid4())
sDateTimeKey=BirthDateZoneStr.replace(' ','-').replace(':','-')
TimeLine=[('ZoneBaseKey', ['UTC']),
         ('IDNumber', [IDZoneNumber]),

         ('DateTimeKey', [sDateTimeKey]), ('UTCDateTimeValue',
         [BirthDateZoneUTC]), ('Zone', [BirthZone]),
         ('DateTimeValue', [BirthDateStr])]
TimeFrame = pd.DataFrame.from_dict(dict(TimeLine))
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
sTable = 'Hub-Time-Gunnarsson'
print('Storing :',sDatabaseName,'\n Table:',sTable)
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace") sTable =
'Dim-Time-Gunnarsson' TimeHubIndex.to_sql(sTable, conn3,
```

```
if_exists="replace")
TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']]
TimeSatelliteIndex=TimeSatellite.set_index(['IDNumber'],inplace=False)
BirthZoneFix=BirthZone.replace(' ','-').replace('/','-')
sTable = 'Satellite-Time-' + BirthZoneFix + '-Gunnarsson'
print('Storing :',sDatabaseName,'\n Table:',sTable)
TimeSatelliteIndex.to_sql(sTable, conn2, if_exists="replace") sTable =
'Dim-Time-' + BirthZoneFix + '-Gunnarsson'
TimeSatelliteIndex.to_sql(sTable, conn3, if_exists="replace")
print('Person Category') FirstName =
'Guðmundur' LastName = 'Gunnarsson'
print('Name:',FirstName,LastName) print('Birth
Date:',BirthDateLocal) print('Birth
Zone:',BirthZone)
print('UTC Birth Date:',BirthDateZoneStr)
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('IDNumber', [IDPersonNumber]),
        ('FirstName', [FirstName]),
        ('LastName', [LastName]),
        ('Zone', ['UTC']),
        ('DateTimeValue', [BirthDateZoneStr])] PersonFrame =
pd.DataFrame.from_dict(dict(PersonLine))
TimeHub=PersonFrame
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)

sTable = 'Hub-Person-Gunnarsson'
print('Storing :',sDatabaseName,'\n Table:',sTable)
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace") sTable =
'Dim-Person-Gunnarsson' TimeHubIndex.to_sql(sTable, conn3,
if_exists="replace")
```

OUTPUT

```
##################################
Working Base : C:/VKHCG  using  win32

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

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

##################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Satellite-Time-Atlantic-Reykjavik-Gunnarss

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

##################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Person-Gunnarsson
```

Practical 8 : Organize-Horizontal

```
import sys import os
import pandas as pd import sqlite3 as sq
 if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
Company='01-Vermeulen' sDataWarehouseDir=Base + '/99-DW'
 if not os.path.exists(sDataWarehouseDir):
     os.makedirs(sDataWarehouseDir)
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn1 =
sq.connect(sDatabaseName)
sDatabaseName=sDataWarehouseDir + '/datamart.db' conn2 =
sq.connect(sDatabaseName)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT *
FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT
PersonID,\
        Height,\ Weight,\ bmi,\
        Indicator\
   FROM [Dim-BMI]\ WHERE \
   Height > 1.5 \ and Indicator = 1\
   ORDER BY \
        Height,\ Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
sTable = 'Dim-BMI-Horizontal'
print('Storing :',sDatabaseName,'\n Table:',sTable)
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI-Horizontal'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT *
FROM [Dim-BMI];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
print('Full Data Set (Rows):', PersonFrame0.shape[0]) print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0]) print('Horizontal Data Set (Columns):',
PersonFrame2.shape[1])
```

OUTPUT

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

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

Practical 8 : Organize-Vertical

```
import sys import os
import pandas as pd import sqlite3 as sq
 if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
Company='01-Vermeulen' sDataWarehouseDir=Base + '/99-DW'
 if not os.path.exists(sDataWarehouseDir):
     os.makedirs(sDataWarehouseDir)
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn1 =
sq.connect(sDatabaseName)
sDatabaseName=sDataWarehouseDir + '/datamart.db' conn2 =
sq.connect(sDatabaseName)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT *
FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
       Height,\ Weight,\ Indicator\
  FROM [Dim-BMI];"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
sTable = 'Dim-BMI-Vertical')
print('Storing :',sDatabaseName,'\n Table:',sTable)
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT *
FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
print('Full Data Set (Rows):', PersonFrame0.shape[0]) print('Full Data Set
(Columns):', PersonFrame0.shape[1])
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0]) print('Horizontal Data Set
(Columns):', PersonFrame2.shape[1])
```

OUTPUT

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

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

Practical 8 : Organize-island

```
import sys import os
import pandas as pd import sqlite3 as sq
 if sys.platform == 'linux': Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
Company='01-Vermeulen' sDataWarehouseDir=Base + '/99-DW'
 if not os.path.exists(sDataWarehouseDir):
     os.makedirs(sDataWarehouseDir)
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn1 =
sq.connect(sDatabaseName)
sDatabaseName=sDataWarehouseDir + '/datamart.db' conn2 =
sq.connect(sDatabaseName)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT *
FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
       Height,\ Weight,\ Indicator\
   FROM [Dim-BMI]\
   WHERE Indicator > 2\ ORDER BY \
       Height,\ Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
sTable = 'Dim-BMI-Vertical'
print('Storing :',sDatabaseName,'\n Table:',sTable)
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
print('Full Data Set (Rows):', PersonFrame0.shape[0]) print('Full Data Set
(Columns):', PersonFrame0.shape[1])
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0]) print('Horizontal Data Set
(Columns):', PersonFrame2.shape[1])
```

OUTPUT

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

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

Practical 8 : Organize-secure-vault

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

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn1 =
sq.connect(sDatabaseName)
sDatabaseName=sDataWarehouseDir + '/datamart.db' conn2 =
sq.connect(sDatabaseName)
sTable = 'Dim-BMI'




print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT *

FROM [Dim-BMI];"

PersonFrame0=pd.read_sql_query(sSQL, conn1)
STable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \




        Height,\ Weight,\ Indicator,\
        CASE Indicator\


WHEN 1 THEN 'Pip'\
        WHEN 2 THEN 'Norman'\ WHEN 3 THEN
        'Grant'\ ELSE 'Sam'\
        END AS Name\ FROM [Dim-BMI]\
   WHERE Indicator > 2\ ORDER BY \
        Height,\ Weigh
```

```
PersonFrame1=pd.read_sql_query(sSQL, conn1)
        DimPerson=PersonFrame1

DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)

   sTable = 'Dim-BMI-Secure'
      print('Storing :',sDatabaseName,'\n Table:',sTable) DimPersonIndex.to_sql(sTable,

conn2, if_exists="replace")
      sTable = 'Dim-BMI-Secure'
```

OUTPUT

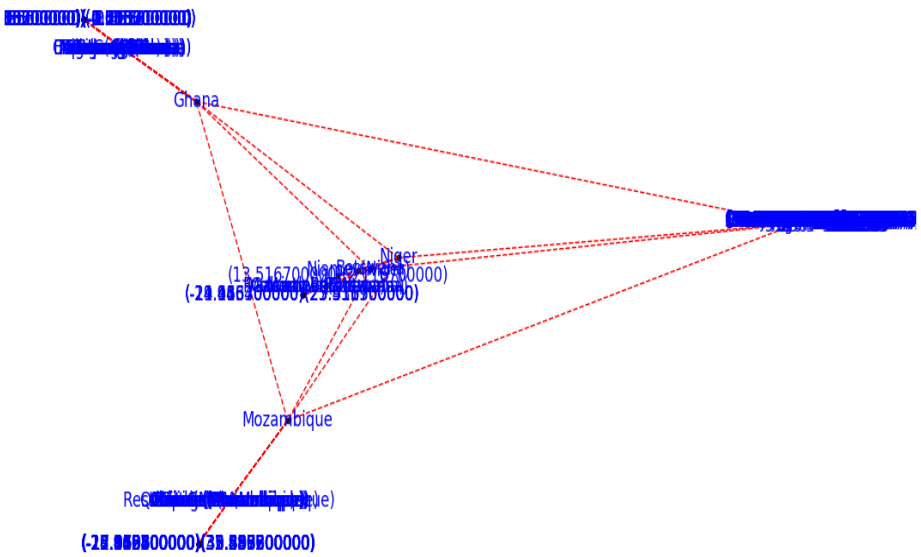
```
################################
Working Base : C:/VKHCG  using  win32
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

################################
Storing : C:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI-Secure
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Secure
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
Horizontal Data Set (Rows): 692
Horizontal Data Set (Columns): 4
Only Sam Data
   Indicator  Height  Weight Name
0          4     1.0      35  Sam
1          4     1.0      40  Sam
2          4     1.0      45  Sam
3          4     1.0      50  Sam
4          4     1.0      55  Sam
```

Practical No.9

A) Generating Reports

```
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
pd.options.mode.chained_assignment = None
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/VKHCG'
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv'
sOutputFileName1='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml'
sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png'
Company='01-Vermeulen'
sFileName=Base + '/' + Company + '/' + sInputFileName
CustomerDataRaw=pd.read_csv(sFileName, header=0, low_memory=False, encoding="latin-1")
CustomerData=CustomerDataRaw.head(100)
G=nx.Graph()
for i in range(CustomerData.shape[0]):
   for j in range(CustomerData.shape[0]):
      Node0=CustomerData['Customer_Country_Name'][i]
      Node1=CustomerData['Customer_Country_Name'][j]
      if Node0 != Node1:
         G.add_edge(Node0,Node1)
for i in range(CustomerData.shape[0]):
   Node0=CustomerData['Customer_Country_Name'][i]
   Node1=CustomerData['Customer_Place_Name'][i] + '('+ CustomerData['Customer_Country_Name'][i] + ')'
   Node2='('+ "{:.9f}".format(CustomerData['Customer_Latitude'][i]) + ')('+
"{:.9f}".format(CustomerData['Customer_Longitude'][i]) + ')'
   if Node0 != Node1:
      G.add_edge(Node0,Node1)
   if Node1 != Node2:
      G.add_edge(Node1,Node2)
sFileName=Base + '/' + Company + '/' + sOutputFileName1
nx.write_gml(G, sFileName)
sFileName=Base + '/' + Company + '/' + sOutputFileName2
plt.figure(figsize=(25, 25))
pos=nx.spectral_layout(G, dim=2)
nx.draw_networkx_nodes(G, pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos, edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G, pos, font_size=12, font_family='sans-serif', font_color='b')
plt.axis('off')
plt.savefig(sFileName, dpi=600)
plt.show()
```

Output:


Ghana
Niger
Mozambique

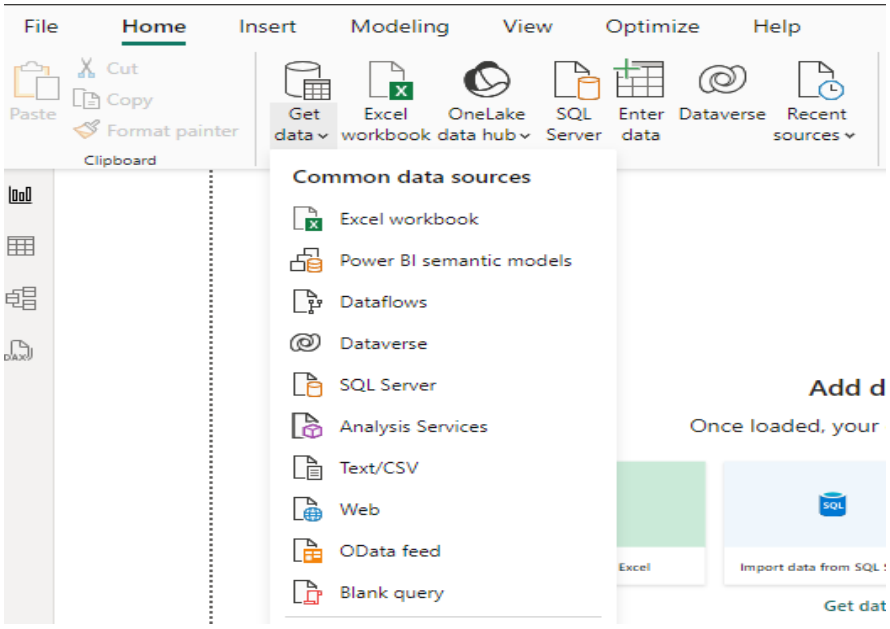
Practical No. 10

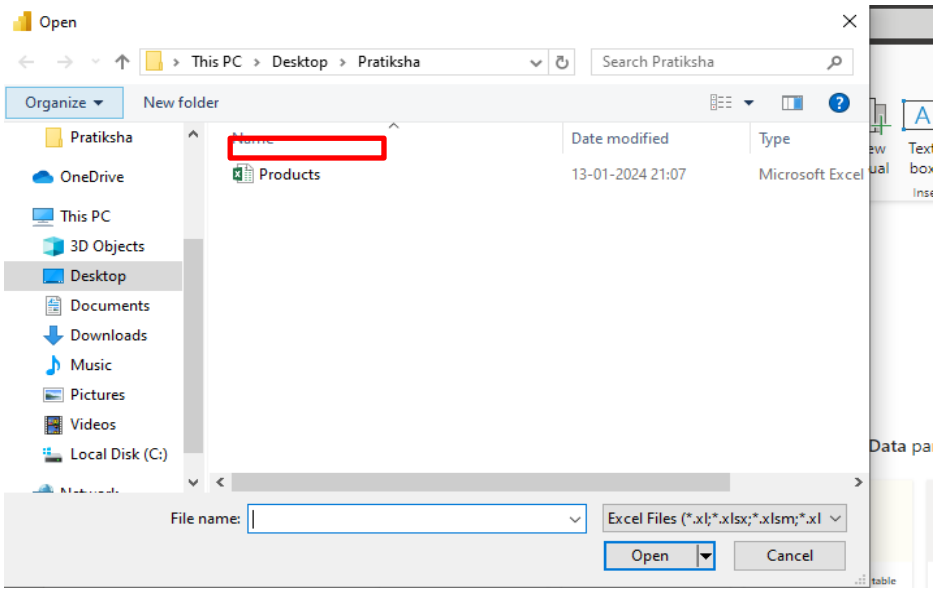
Data Visualization with power bi

Case Study : Sales Data

Step 1: connect to an Excel Workbook

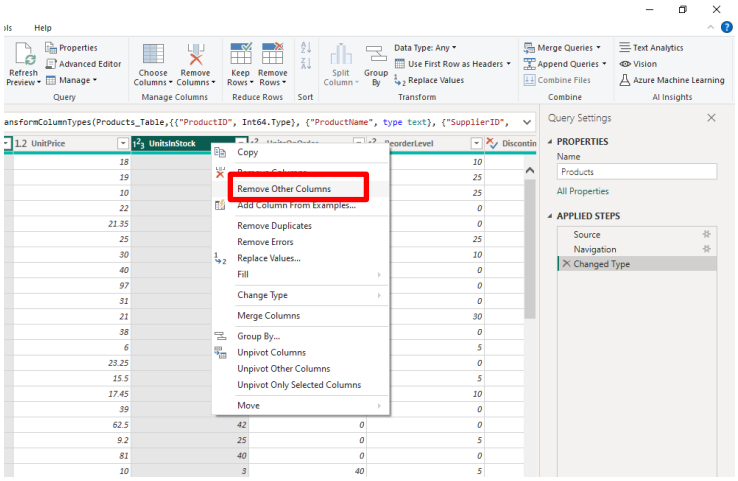
1. Launch power Bi Desktop.
2. From the Home Ribbon, Select Get Data Select Excel Workbook .
3. In the Open File Dialog Box, Select the Product.xlsx file.



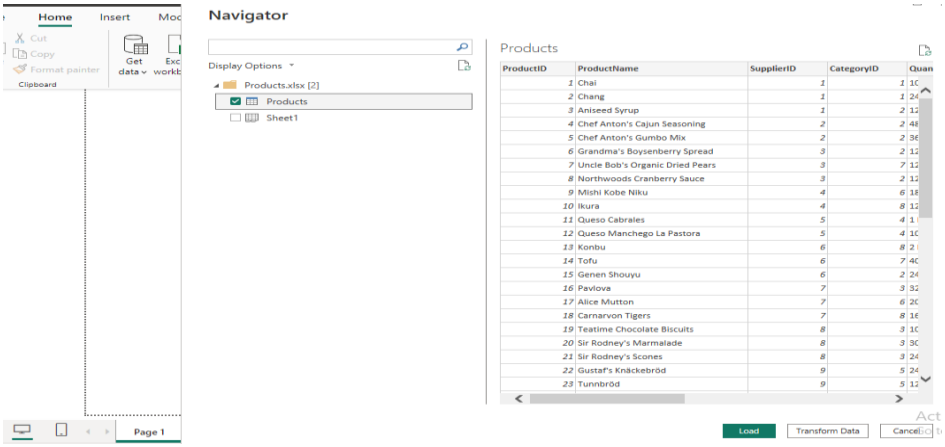


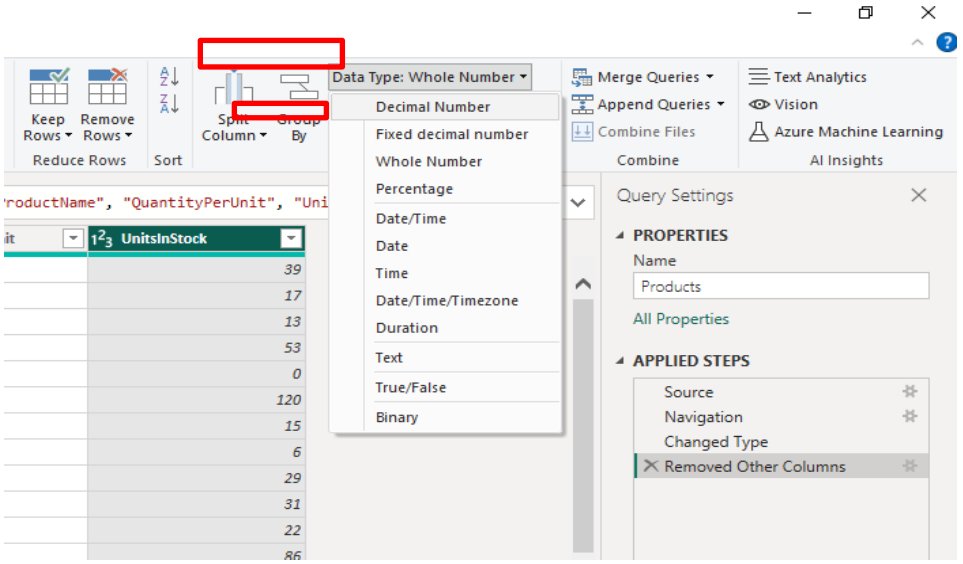
4. In the Open File Dialog Box, Select the Product.xlsx file.

5. Click on Products Check Box. & We Will see the product Table. Select The Transform Data.
6. In Query Editor , Select the ProductID, ProductName, QuantityPerUnit, and unitsInStock Columns. (Use Ctrl + Click to select more than one column).



7. Right Click on Column Header and Click Remove Other Columns.

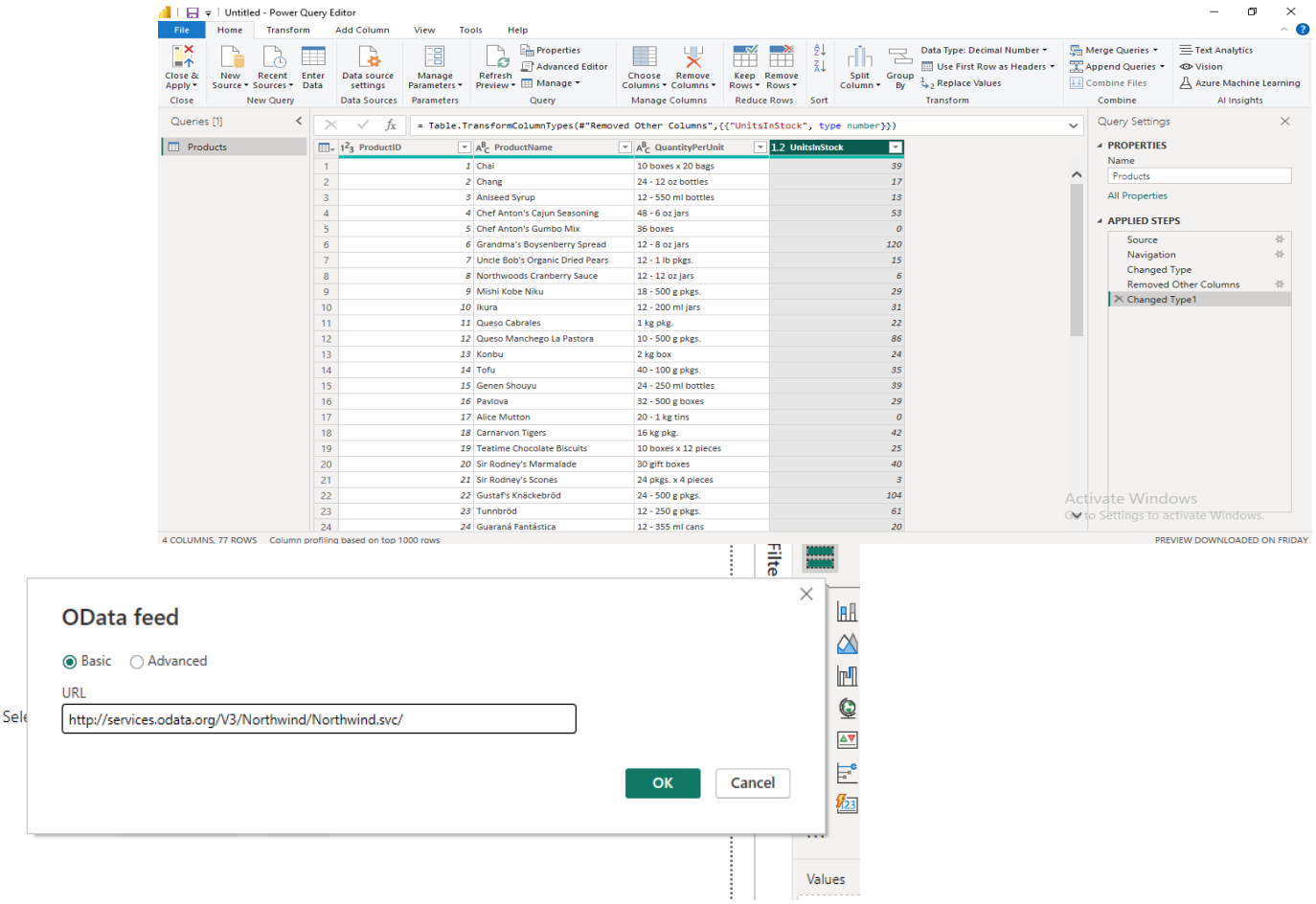




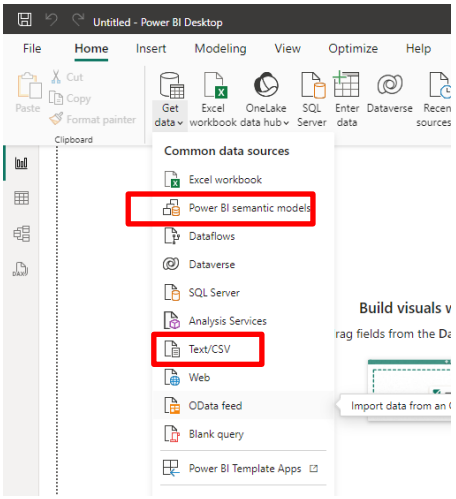
8. Select Close & Apply from Home Ribbon.

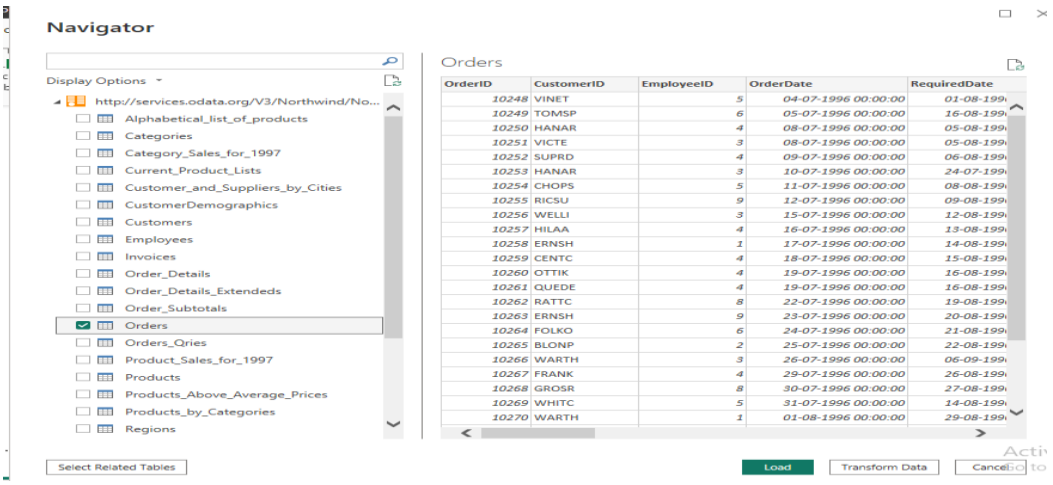
9. Another Window on Select Get Data & Select the OData feed. And Copy the link given below. & Paste it to OData feed URL Box and Click Ok.

http://services.odata.org/V3/Northwind/Northwind.svc/

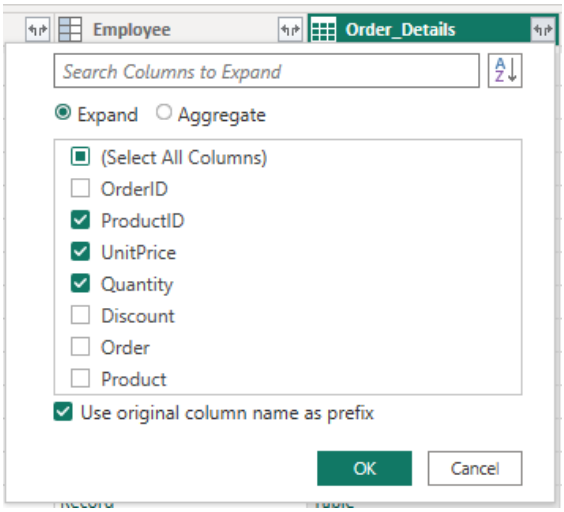


10. We will see the below screen like this then select from navigator orders checkbox. & Click on Transform Data.

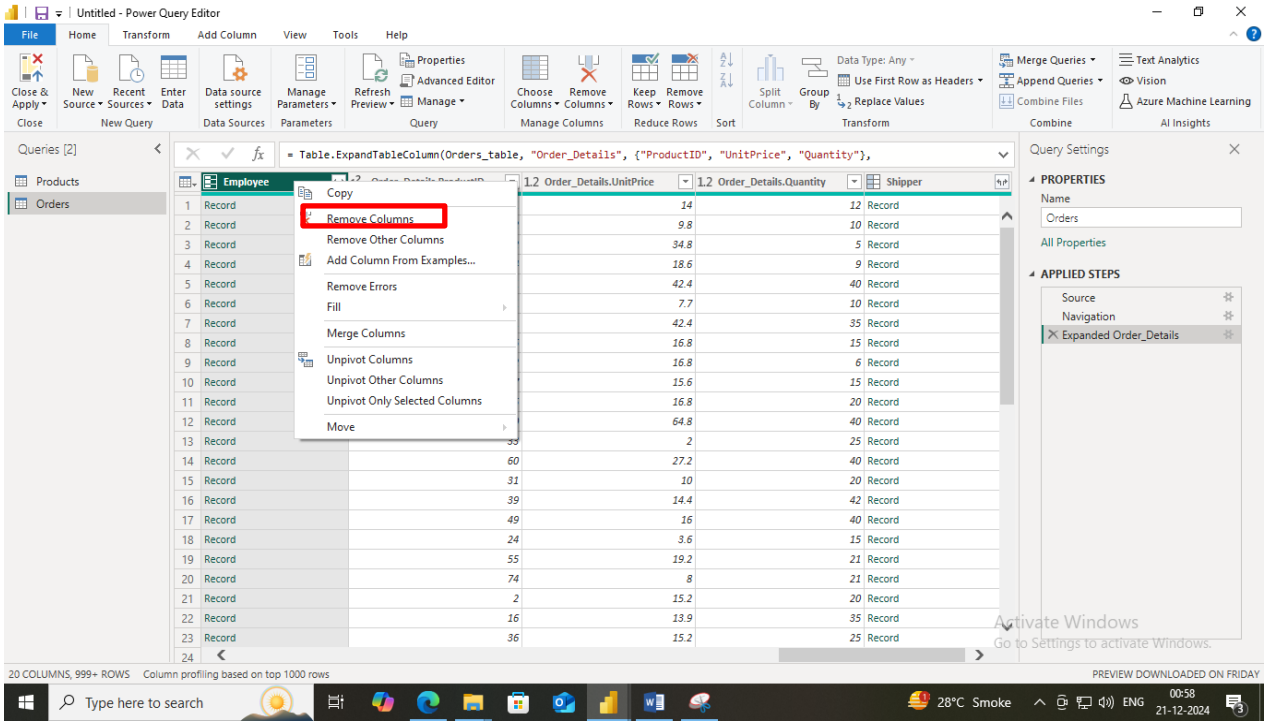




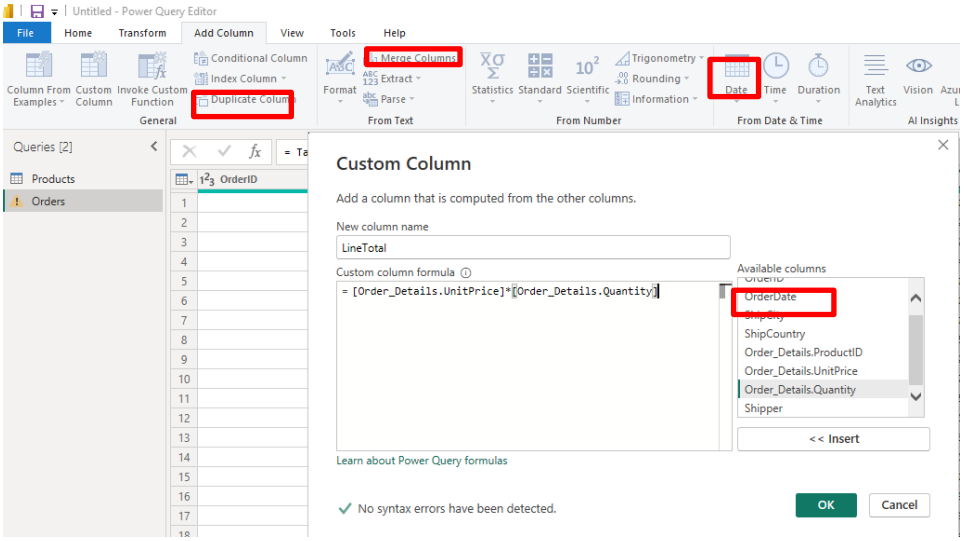
11. Expand the Order_Details column & select the ProductID, UnitPrice, Quantity & Click OK.



12. Remove other Column to only display column of interest, In this step you remove all Column except OrderID, OrderDate, ShipCity, Order_Details.ProductID, Order_Details.Unitprice, Order_Details.Quantity, Shipper columns. & Remove Columns.

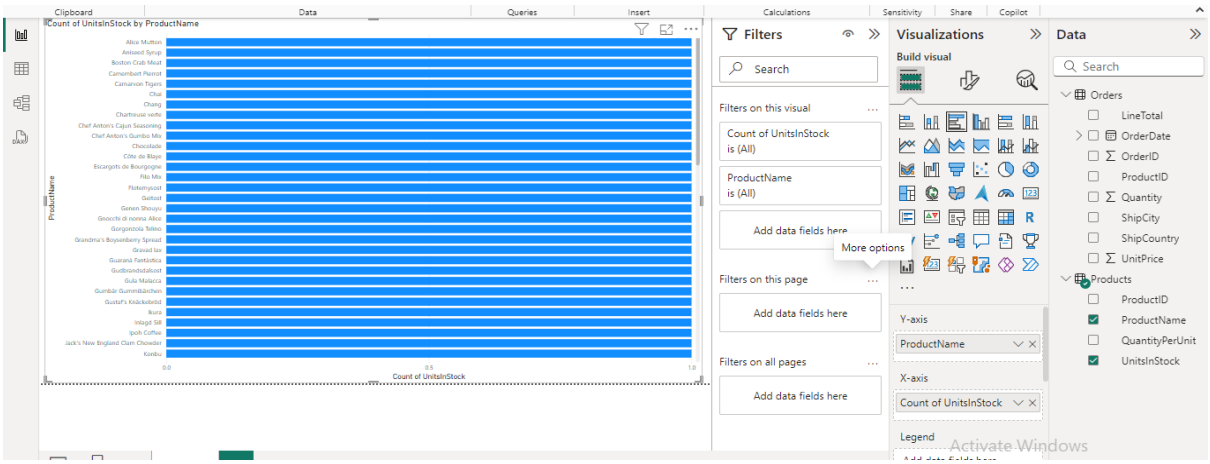


13. From Add Column Ribbon Select Custom Column. Add New Name in new Column name LineTotal. From Available column Select order_Details.Unitprice and Click insert Add "*" and select Order_Details.Quantity and insert Ok. We Will see the a New Column Name LineTotal Appears.



14. In Query Editor, drag the LineTotal Column to the left , After ShipCountry. Double Click on Order_Details.ProductID, Order_Details.Unitprice, Order_Details.Quantity change name to Only ProductID, Unitprice, Quantity.

15. From Home Ribbon , Select Close and apply. We Will get new Window of Power Bi. Select From Data Paneel From Products select ProductName And UnitInStock. If output is not seen then Change X-axis and Y-axis from Visualizations.



16. For Orders Select Map from Visualizations. And From data Column Select From Orders Select LineTotal And ShipCity.