PRACTICAL 1

AIM : - CREATING DATA MODEL USING CASSANDRA

THEORY: -

Apache Cassandra is a large-scale distributed database supporting multi data centre replication for availability, durability, and performance. The standard Apache Cassandra open-source version works just as well, minus some extra management itdoes not offer as standard.

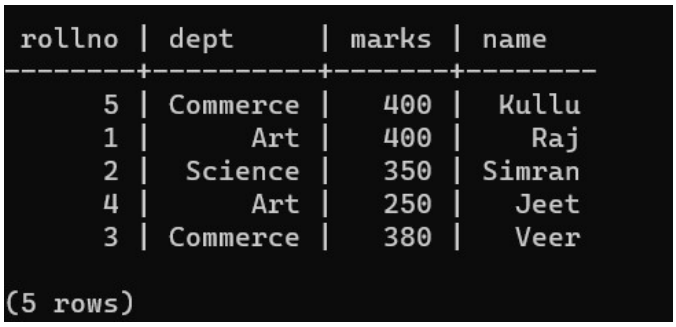
1. **Cassandra Create Keyspace**
   To communicate with Cassandra the Cassandra Query Language (CQL) is used. A keyspace is an object working similar to an RDBMS database. It holdscolumn families, the strategy used in the keyspace, indexes, user-defined types,replication factor, data centre awareness, etc.
2. **Insert :** This command is used to insert the data in the table.
3. **Update :** This command is used to update a data in the table.
4. **Delete :** This command is used to delete data from the table
5. **Alter Table :** To alter an existing table, the ALTER TABLE command isused. This command can be used to add a column or to Drop a column.
6. **Truncate Table :** To truncate a table in Cassandra, the TRUNCATE command is used. Thus, all the rows are deleted permanently from the table,but the table structure exists.
7. **Drop Table :** This command is used to drop the entire table from the database.

Code :-

cqlsh> create keyspace University with replication={'class':'SimpleStrategy','replication_factor':1}; cqlsh> Use University;
cqlsh:university> create table student(RollNO int Primary key,Name text, dept text,Marks int);

```
cqlsh:university> Insert into
student(RollNo,Name,dept,Marks)values(01,'Raj','Art',400);
cqlsh:university> Insert into
student(RollNo,Name,dept,Marks)values(02,'Simran','Science',350);
cqlsh:university> Insert into
student(RollNo,Name,dept,Marks)values(03,'Veer','Commerce',380);
cqlsh:university> Insert into
student(RollNo,Name,dept,Marks)values(04,'Jeet','Art',250);
cqlsh:university> Insert into
student(RollNo,Name,dept,Marks)values(05,'Kullu','Commerce',400);
```

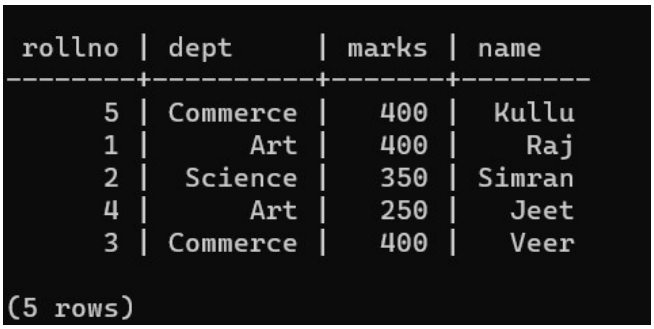
cqlsh:university> select*from student;



```
 rollno | dept     | marks | name
--------+----------+-------+--------
      5 | Commerce |   400 |  Kullu
      1 |      Art |   400 |    Raj
      2 |  Science |   350 | Simran
      4 |      Art |   250 |   Jeet
      3 | Commerce |   380 |   Veer

(5 rows)
```

cqlsh:university> update student set Marks=400 where RollNo=03;

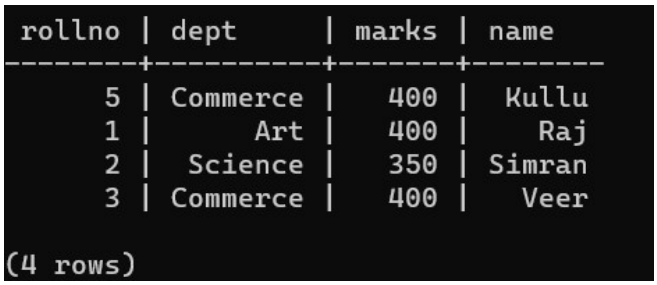
cqlsh:university> select*from student;



```
 rollno | dept     | marks | name
--------+----------+-------+--------
      5 | Commerce |   400 |  Kullu
      1 |      Art |   400 |    Raj
      2 |  Science |   350 | Simran
      4 |      Art |   250 |   Jeet
      3 | Commerce |   400 |   Veer

(5 rows)
```

cqlsh:university> Delete from student where RollNo=04;

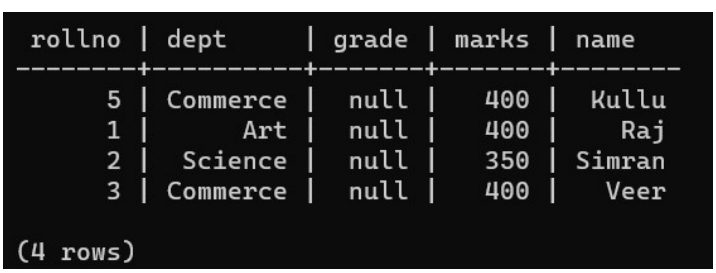
cqlsh:university> select*from student;



```
 rollno | dept     | marks | name
--------+----------+-------+--------
      5 | Commerce |   400 |  Kullu
      1 |      Art |   400 |    Raj
      2 |  Science |   350 | Simran
      3 | Commerce |   400 |   Veer

(4 rows)
```

cqlsh:university> Alter table student add Grade text;

cqlsh:university> select*from student;



```
 rollno | dept     | grade | marks | name
--------+----------+-------+-------+--------
      5 | Commerce |  null |   400 |  Kullu
      1 |      Art |  null |   400 |    Raj
      2 |  Science |  null |   350 | Simran
      3 | Commerce |  null |   400 |   Veer

(4 rows)
```

cqlsh:university> Insert into
student(RollNo,Name,dept,Marks,Grade)values(05,'Kullu','Commerce',400,'A');
cqlsh:university> Insert into
student(RollNo,Name,dept,Marks,Grade)values(04,'Jeet','Art',250,'C'); cqlsh:university>
Insert into
student(RollNo,Name,dept,Marks,Grade)values(03,'Veer','Commerce',380,'B');
cqlsh:university> Insert into
student(RollNo,Name,dept,Marks,Grade)values(02,'Simran','Science',350,'B');
cqlsh:university> Insert into
student(RollNo,Name,dept,Marks,Grade)values(01,'Raj','Art',400,'A'); cqlsh:university>
select*from student;

```
 rollno | dept     | grade | marks | name
--------+----------+-------+-------+--------
      5 | Commerce |     A |   400 |  Kullu
      1 |      Art |     A |   400 |    Raj
      2 |  Science |     B |   350 | Simran
      4 |      Art |     C |   250 |   Jeet
      3 | Commerce |     B |   380 |   Veer

(5 rows)
```

cqlsh:university> Truncate table student;

cqlsh:university> select*from student;

```
 rollno | dept | grade | marks | name
--------+------+-------+-------+------

(0 rows)
```

cqlsh:university> Drop table student;

cqlsh:university> select*from student;

```
cqlsh:university> Drop table student;
cqlsh:university> select*from student;
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured table student"
```

PRACTICAL 2

AIM :- CONVERSION FROM DIFFERENT FORMATS TO HORUS FORMAT

A. Text delimited CSV to HORUS format.

Write a Python / R program for text delimited csv to horus format in data science.

Code :-

```
# Utility Start CSV to HORUS ================================
# Standard Tools
#============================================================
import pandas as pd
# Input Agreement ===========================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
# Processing Rules ==========================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ==========================================
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
```

Output :-

```
========== RESTART: C:\Msc-IT-Seml_Practicals\DS practicals\p2A.py ==========
Input Data Values ===================================
                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0                 Afghanistan         AF         AFG        4
1               Aland Islands         AX         ALA      248
2                     Albania         AL         ALB        8
3                     Algeria         DZ         DZA       12
4              American Samoa         AS         ASM       16
..                       ...        ...         ...      ...
242  Wallis and Futuna Islands         WF         WLF      876
243             Western Sahara         EH         ESH      732
244                      Yemen         YE         YEM      887
245                     Zambia         ZM         ZMB      894
246                   Zimbabwe         ZW         ZWE      716

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

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

B. XML to HORUS Format

Write a Python / R program for xml to horus format in data science.

Code :-

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

```
            record[subchild.tag] = subchild.text
        all_records.append(record)
    return pd.DataFrame(all_records)
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()
print('=================================================')
print('Input Data Values ===============================')
print('=================================================')
print(InputData)
print('=================================================')
#=================================================================
# Processing Rules ===============================================
#=================================================================
ProcessDataXML=InputData
# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('=================================================')
print('Process Data Values =============================')
print('=================================================')
print(ProcessData)
print('=================================================')
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=================================================')
print('XML to HORUS - Done')
print('=================================================')
# Utility done ===================================================
```

Output :-

```
========== RESTART: C:\Msc-IT-Sem1_Practicals\DS practicals\p2B.py ==========
=====================================================
Input Data Values ===================================
=====================================================
Squeezed text (707 lines).
=====================================================
=====================================================
Process Data Values =================================
=====================================================
                              CountryName
CountryNumber
716                              Zimbabwe
894                                Zambia
887                                 Yemen
732                        Western Sahara
876            Wallis and Futuna Islands
...                                  ...
16                         American Samoa
12                                Algeria
8                                 Albania
248                         Aland Islands
4                             Afghanistan

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

C. JSON to HORUS Format

Write a Python / R program for json to horus format in data science.

Code :-

```
# Utility Start JSON to HORUS ==================================
# Standard Tools
#===============================================================
import pandas as pd
# Input Agreement ==============================================
sInputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/Country_Code.json'
InputData=pd.read_json(sInputFileName, orient='index', encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
# Processing Rules ============================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =================================')
print(ProcessData)
```

```
print('=============================================================')
# Output Agreement ============================================
OutputData=ProcessData
sOutputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('JSON to HORUS - Done')
# Utility done ================================================
```

Output :-

```
=========== RESTART: C:\Msc-IT-Seml_Practicals\DS practicals\p2C.py ==========
Input Data Values ===================================
                        Country ISO-2-CODE ISO-3-Code  ISO-M49
0                   Afghanistan         AF        AFG        4
1                Aland Islands         AX        ALA      248
2                       Albania         AL        ALB        8
3                       Algeria         DZ        DZA       12
4                American Samoa         AS        ASM       16
..                          ...        ...        ...      ...
242  Wallis and Futuna Islands         WF        WLF      876
243              Western Sahara         EH        ESH      732
244                       Yemen         YE        YEM      887
245                      Zambia         ZM        ZMB      894
246                    Zimbabwe         ZW        ZWE      716

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

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

D. MySQL Database to HORUS Format

Write a Python / R program for MySQL database to horus format in data science.

Code :-

```
# Utility Start Database to HORUS ================================
# Standard Tools
#=============================================================
import pandas as pd
import sqlite3 as sq
# Input Agreement ============================================
sInputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)
print('Input Data Values ===================================')
```

```
print(InputData)
print('=====================================================')
# Processing Rules ============================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code',  axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values ===================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ============================================
OutputData=ProcessData
sOutputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('Database to HORUS - Done')
# Utility done ================================================
```

Output :-

```
>>>
========== RESTART: C:\Users\admin\Desktop\MSC.Practical\DS\prac2d.py ==========
Input Data Values ===================================
     index                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0        0                Afghanistan         AF        AFG        4
1        1              Aland Islands         AX        ALA      248
2        2                    Albania         AL        ALB        8
3        3                    Algeria         DZ        DZA       12
4        4             American Samoa         AS        ASM       16
..     ...                        ...        ...        ...      ...
242    242  Wallis and Futuna Islands         WF        WLF      876
243    243             Western Sahara         EH        ESH      732
244    244                      Yemen         YE        YEM      887
245    245                     Zambia         ZM        ZMB      894
246    246                   Zimbabwe         ZW        ZWE      716

[247 rows x 5 columns]
=====================================================
Process Data Values ===================================
              index                CountryName ISO-2-CODE ISO-3-Code
CountryNumber
716             246                   Zimbabwe         ZW        ZWE
894             245                     Zambia         ZM        ZMB
887             244                      Yemen         YE        YEM
732             243             Western Sahara         EH        ESH
876             242  Wallis and Futuna Islands         WF        WLF
...             ...                        ...        ...      ...
16                4             American Samoa         AS        ASM
12                3                    Algeria         DZ        DZA
8                 2                    Albania         AL        ALB
248               1              Aland Islands         AX        ALA
4                 0                Afghanistan         AF        AFG

[247 rows x 4 columns]
=====================================================
Database to HORUS - Done
>>>
```

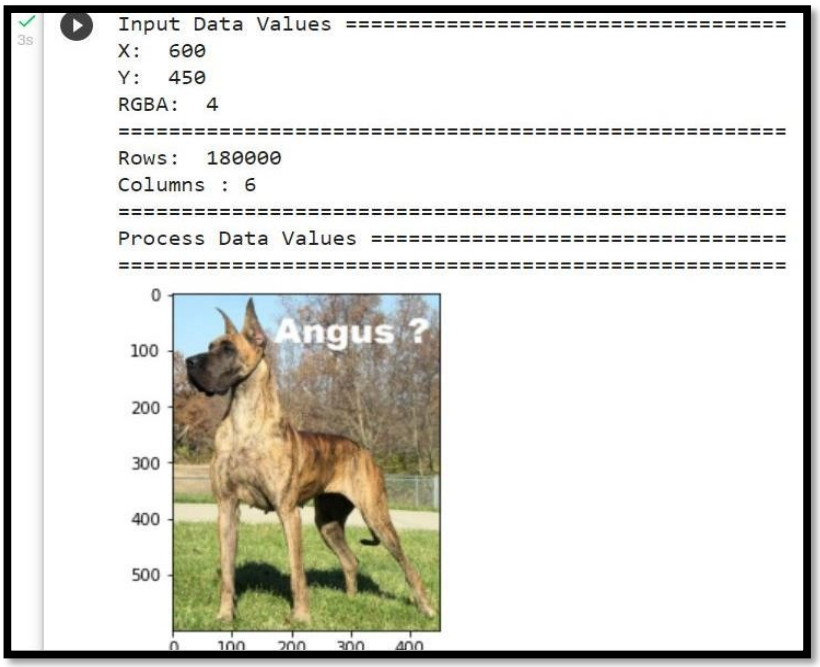
E. Picture (JPEG) to HORUS Format

Write a Python / R program for picture to horus format in data science.

Code :-

```
# Utility Start Picture to HORUS ================================
# Standard Tools
#=============================================================
from scipy.misc import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Input Agreement ============================================
sInputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/Angus.jpg'
InputData = imread(sInputFileName, flatten=False, mode='RGBA')
print('Input Data Values ===================================')
print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])
print('=====================================================')
# Processing Rules ===========================================
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']
ProcessData.columns=sColumns
ProcessData.index.names =['ID']
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
print('=====================================================')
print('Process Data Values ================================')
print('=====================================================')
plt.imshow(InputData)
plt.show()
print('=====================================================')
# Output Agreement ===========================================
OutputData=ProcessData
print('Storing File')
sOutputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Picture to HORUS - Done')
print('=====================================================')
```

Output :-



F. Video to HORUS Format

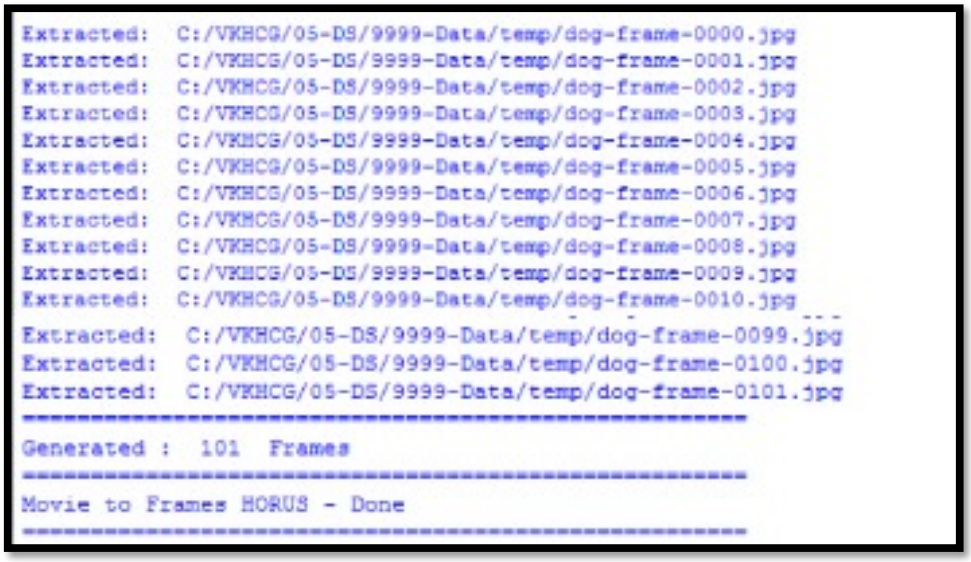
Write a Python / R program for video to horus format in data science.

Code :-

```
# Utility Start Movie to HORUS (Part 1) ======================
# Standard Tools
#=============================================================
import os
import shutil
import cv2
#=============================================================
sInputFileName='C:/Users/Abdul Razzaque/Desktop/practical-data-science/VKHCG/05-DS/9999-Data/Dog.mp4'
sDataBaseDir='C:/Users/Abdul Razzaque/Desktop/practical-data-science/VKHCG/05-DS/9999-Data/temp'
if os.path.exists(sDataBaseDir):
 shutil.rmtree(sDataBaseDir)
if not os.path.exists(sDataBaseDir):
 os.makedirs(sDataBaseDir)
print('=============================================================')
print('Start Movie to Frames')
print('=============================================================')
vidcap = cv2.VideoCapture(sInputFileName)
success,image = vidcap.read()
count = 0
while success:
 success,image = vidcap.read()
```

```
sFrame=sDataBaseDir + str('/dog-frame-' + str(format(count, '04d'))+ '.jpg')
print('Extracted: ', sFrame)
cv2.imwrite(sFrame, image)
if os.path.getsize(sFrame) == 0:
 count += -1
os.remove(sFrame)
print('Removed: ', sFrame)
if cv2.waitKey(10) == 27: # exit if Escape is hit
 count += 1
print('=====================================================')
```

Output :-



G. Audio to HORUS Format

Write a Python / R program for audio to horus format in data science.

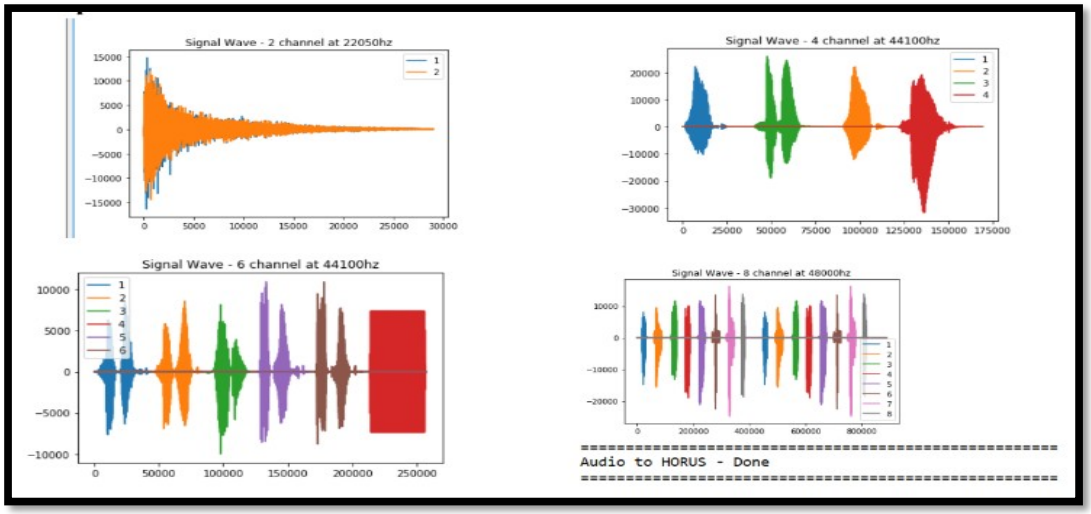
Code :-

```
# Utility Start Audio to HORUS ================================
# Standard Tools
#=============================================================
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/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/2ch-sound.wav'
print('=========================================================')
print('Processing : ', sInputFileName)
print('=========================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('=========================================================')
print('Processing : ', sInputFileName)
print('=========================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("4 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#============================================================
sInputFileName='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-
science/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('=========================================================')
print('Processing : ', sInputFileName)
print('=========================================================')
```

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

Output :-



PRACTICAL 3

AIM :- UTILITIES AND AUDITING

A. Fixers Utilities

Write a Python / R program for basic fixers utilities in data science.

Code :-

```
#---------------------------- Program to Demonstrate Fixers utilities --------------------
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print('#1 Removing leading or lagging spaces from a data entry');
baddata = " 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)
```

Output :-

```
========== RESTART: C:\Msc-IT-Sem1_Practicals\DS practicals\p3A.py ==========
#1 Removing leading or lagging spaces from a data entry
>  Data Science with too many spaces is bad!!!  <
> Data Science with too many spaces is bad!!! <
#2 Removing nonprintable characters from a data entry
Bad Data :  Data
Clean Data :  D a t a S c i e n c e   w i t h   f u n n y   c h a r a c t e r s   i s   b a d ! ! !
# 3 Reformatting data entry to match specific formatting criteria.
Bad Data :  2019-10-31
Good Data :  31 October 2019
```

B. Data Binning or Bucketing

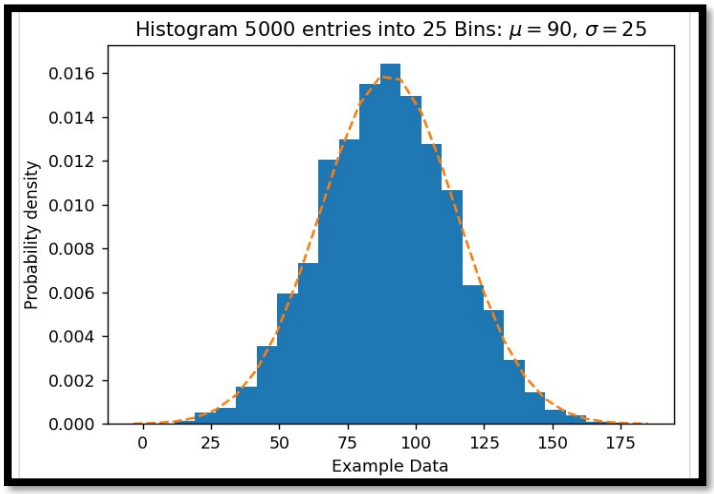
Write a Python / R program for basic data binning or bucketing in data science.

Binning is a data pre-processing technique used to reduce the effects of minor observation errors. Statistical data binning is a way to group a number of more or less continuous values into a smaller number of "bins."

Code :-

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

Output :-



C . Averaging of Data

Write a Python / R program for basic averaging of data in data science.

The use of averaging of features value enables the reduction of data volumes in a control fashion to improve effective data processing.
C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Mean.py

Code :-

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

Output :-

```
============ RESTART: C:\Msc-IT-Sem1_Practicals\DS practicals\p3C.py ===========
################################
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

Write a Python / R program for basic outlier detection in data science.

Outliers are data that is so different from the rest of the data in the data set that it may be caused by an error in the data source. There is a technique called outlier detection that, with good data science, will identify these outliers.
C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py

Code :-

```
# -*- coding: utf-8 -*-
########################################################################
import pandas as pd
########################################################################
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base)
print('##################################')
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data')
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) & (AllData.Latitude<=UpperBound)]
print(OutliersNot)
########################################################################
```

Output :-

```
========== RESTART: C:\Msc-IT-Seml_Practicals\DS practicals\p3D.
###################################
Working Base : C:/VKHCG
###################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
All Data
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
...      ...        ...       ...
3434      GB     London   51.5092
3435      GB     London   51.5092
3436      GB     London   51.5163
3437      GB     London   51.5085
3438      GB     London   51.5136

[1502 rows x 3 columns]
Outliers
Higher than  51.512635507867415
     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
```

```
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]
```

E. Logging

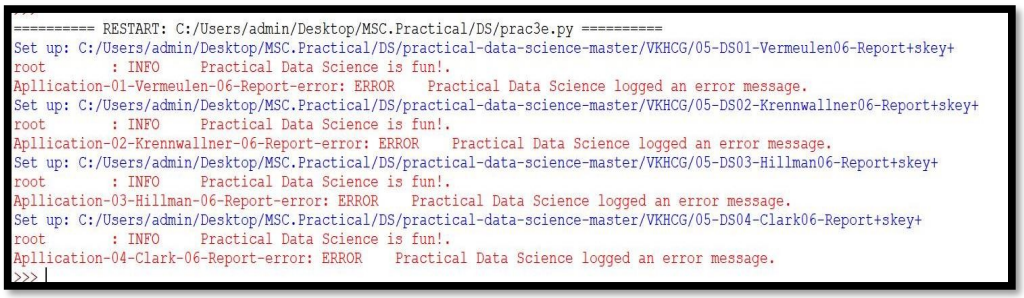
Write a Python / R program for basic logging in data science.

Code :-

```
import sys
import os
import logging
import uuid
import shutil
import time
Base='C:/VKHCG'
sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04-Clark']
sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-Organise','06-Report']
sLevels=['debug','info','warning','error']
for sCompany in sCompanies:
    sFileDir=Base + '/' + sCompany
    if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
    for sLayer in sLayers:
        log = logging.getLogger() # root logger
        for hdlr in log.handlers[:]: # remove all old handlers
            log.removeHandler(hdlr)
#-----------------------------------------------------------------------------------------------
        sFileDir=Base + '/' + sCompany + '/' + sLayer + '/Logging'
        if os.path.exists(sFileDir):
            shutil.rmtree(sFileDir)
        time.sleep(2)
        if not os.path.exists(sFileDir):
            os.makedirs(sFileDir)
        skey=str(uuid.uuid4())
        sLogFile=Base + '/' + sCompany + '/' + sLayer + '/Logging/Logging_'+skey+'.log'
        print('Set up:',sLogFile)
# set up logging to file - see previous section for more details
        logging.basicConfig(level=logging.DEBUG,format='%(asctime)s %(name)-12s % (levelname)-
8s %(message)s',datefmt='%m-%d %H:%M',filename=sLogFile,filemode='w')
# define a Handler which writes INFO messages or higher to the sys.stderr
        console = logging.StreamHandler()
        console.setLevel(logging.INFO)
# set a format which is simpler for console use
        formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
# tell the handler to use this format
        console.setFormatter(formatter)
# add the handler to the root logger
        logging.getLogger('').addHandler(console)
# Now, we can log to the root logger, or any other logger. First the root...
        logging.info('Practical Data Science is fun!.')
        for sLevel in sLevels:
            sApp='Apllication-'+ sCompany + '-' + sLayer + '-' + sLevel
            logger = logging.getLogger(sApp)
```

```
    if sLevel == 'debug':
        logger.debug('Practical Data Science logged a debugging message.')
    if sLevel == 'info':
        logger.info('Practical Data Science logged information message.')
    if sLevel == 'warning':
        logger.warning('Practical Data Science logged a warning message.')
    if sLevel == 'error':
        logger.error('Practical Data Science logged an error message.')
#----------------------------------------------------------------------------------------
```

Output :-



PRACTICAL 4

AIM :- RETRIEVING DATA

A. Program to retrieve different attributes of data.

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 :-

IP_DATA_COUNTRY_FREQ

Filter

	Country	N
2	GB	1502
3	US	1383
1	DE	677

IP_DATA_COUNTRY_FREQ

Filter

	Country	N
2	GB	1502
3	US	1383
1	DE	677

B. 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('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 :-

```
===== RESTART: C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-IP_DATA_ALL.py =====
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 3562
Columns: 8
### Raw Data Set #########################################
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 #########################################
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!! ################################################
```

ii) 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'
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#####################################################################
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)
#####################################################################
sFileDir=Base + '/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:

```
>>>
=== RESTART: C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-Router-Location.py ===
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
Rows : 150
Columns : 4
### Done!! ############################################
```

(ii) Picking Content for Billboards.

Code:

```
import sys
import os
import pandas as pd
#####################################################################
InputFileName='DE_Billboard_Locations.csv'
OutputFileName='Retrieve_DE_Billboard_Locations.csv'
Company='02-Krennwallner'
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
```

```
print('################################')
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','PlaceName','Latitude','Longitude'])
IP_DATA_ALL.rename(columns={'PlaceName': 'Place_Name'}, inplace=True)
#####################################################################
```

Output:

```
>>>
 RESTART: C:\VKHCG\02-Krennwallner\01-Retrieve\Retrieve-DE-Billboard-Locations.py
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/02-Krennwallner/00-RawData/DE_Billboard_Locations.csv
Rows : 8873
Columns : 4
### Done!! ############################################
```

(iii) 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='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/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','First 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:

```
== RESTART: C:\VKHCG\02-Krennwallner\01-Retrieve\Retrieve-Online-Visitor.py ==
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows : 3562
Columns : 6
Export CSV
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10.csv
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.gz
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.bz2
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.xz
Export JSON
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_split.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_records.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_index.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_columns.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_values.json
```

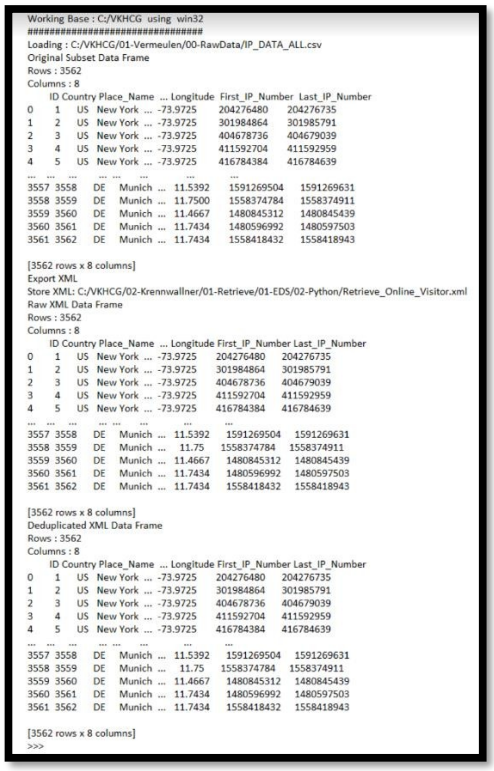
(iv) XML processing.

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('~') + 'C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-
data-science/VKHCG'
else:
 Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
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:



```
Working Base : C:/VKHCG using win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Original Subset Data Frame
Rows : 3562
Columns : 8
      ID Country Place_Name  ... Longitude  First_IP_Number  Last_IP_Number
0      1   US    New York  ... -73.9725      204276480      204276735
1      2   US    New York  ... -73.9725      301984864      301985791
2      3   US    New York  ... -73.9725      404678736      404679039
3      4   US    New York  ... -73.9725      411592704      411592959
4      5   US    New York  ... -73.9725      416784384      416784639
...  ...  ...         ...  ...      ...            ...            ...
3557 3558   DE      Munich  ... 11.5392     1591269504     1591269631
3558 3559   DE      Munich  ... 11.7500     1558374784     1558374911
3559 3560   DE      Munich  ... 11.4667     1480845312     1480845439
3560 3561   DE      Munich  ... 11.7434     1480596992     1480597503
3561 3562   DE      Munich  ... 11.7434     1558418432     1558418943

[3562 rows x 8 columns]
Export XML
Store XML: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.xml
Raw XML Data Frame
Rows : 3562
Columns : 8
      ID Country Place_Name  ... Longitude First_IP_Number Last_IP_Number
0      1   US    New York  ... -73.9725      204276480      204276735
1      2   US    New York  ... -73.9725      301984864      301985791
2      3   US    New York  ... -73.9725      404678736      404679039
3      4   US    New York  ... -73.9725      411592704      411592959
4      5   US    New York  ... -73.9725      416784384      416784639
...  ...  ...         ...  ...      ...            ...            ...
3557 3558   DE      Munich  ... 11.5392     1591269504     1591269631
3558 3559   DE      Munich  ...   11.75     1558374784     1558374911
3559 3560   DE      Munich  ... 11.4667     1480845312     1480845439
3560 3561   DE      Munich  ... 11.7434     1480596992     1480597503
3561 3562   DE      Munich  ... 11.7434     1558418432     1558418943

[3562 rows x 8 columns]
Deduplicated XML Data Frame
Rows : 3562
Columns : 8
      ID Country Place_Name  ... Longitude First_IP_Number Last_IP_Number
0      1   US    New York  ... -73.9725      204276480      204276735
1      2   US    New York  ... -73.9725      301984864      301985791
2      3   US    New York  ... -73.9725      404678736      404679039
3      4   US    New York  ... -73.9725      411592704      411592959
4      5   US    New York  ... -73.9725      416784384      416784639
...  ...  ...         ...  ...      ...            ...            ...
3557 3558   DE      Munich  ... 11.5392     1591269504     1591269631
3558 3559   DE      Munich  ...   11.75     1558374784     1558374911
3559 3560   DE      Munich  ... 11.4667     1480845312     1480845439
3560 3561   DE      Munich  ... 11.7434     1480596992     1480597503
3561 3562   DE      Munich  ... 11.7434     1558418432     1558418943

[3562 rows x 8 columns]
>>>
```

(v) Planning Shipping Rules for Best-Fit International Logistics.

Code:

```
import os
import sys
import pandas as pd
IncoTerm='EXW'
InputFileName='Incoterm_2010.csv'
OutputFileName='Retrieve_Incoterm_' + IncoTerm + '_RuleSet.csv'
Company='03-Hillman'
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#####################################################################
### Import Incoterms
#####################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('#############')
print('Loading :',sFileName)
IncotermGrid=pd.read_csv(sFileName,header=0,low_memory=False)
IncotermRule=IncotermGrid[IncotermGrid.Shipping_Term == IncoTerm]
```

```
print('Rows :',IncotermRule.shape[0])
print('Columns :',IncotermRule.shape[1])
print('############')
print(IncotermRule)
sFileName=sFileDir + '/' + OutputFileName
IncotermRule.to_csv(sFileName, index = False)
print('### Done!! ################################################')
```

Output:

```
>>>
===== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Incoterm-EXW.py =====
################################
Working Base : C:/VKHCG  using  win32
################################
############
Loading : C:/VKHCG/03-Hillman/00-RawData/Incoterm_2010.csv
Rows : 1
Columns : 9
############
  Shipping_Term  Seller Carrier Port_From  ... Port_To Terminal Named_Place  Buyer
0           EXW  Seller   Buyer     Buyer  ...   Buyer    Buyer       Buyer  Buyer

[1 rows x 9 columns]
### Done!! ############################################
```

(vi) FCA—Free Carrier (Named Place of Delivery).

Code:

```
import os
import sys
import pandas as pd
#################################################################
IncoTerm='FCA'
InputFileName='Incoterm_2010.csv'
OutputFileName='Retrieve_Incoterm_' + IncoTerm + '_RuleSet.csv'
Company='03-Hillman'
#################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#################################################################
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#################################################################
### Import Incoterms
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('############')
print('Loading :',sFileName)
IncotermGrid=pd.read_csv(sFileName,header=0,low_memory=False)
IncotermRule=IncotermGrid[IncotermGrid.Shipping_Term == IncoTerm]
print('Rows :',IncotermRule.shape[0])
```

```
print('Columns :',IncotermRule.shape[1])
print('############')
print(IncotermRule)
#########################################################################
sFileName=sFileDir + '/' + OutputFileName
IncotermRule.to_csv(sFileName, index = False)
#########################################################################
print('### Done!! ################################################')
#########################################################################
```

Output:

```
>>>
===== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Incoterm-FCA.py =====
################################
Working Base : C:/VKHCG  using  win32
################################
############
Loading : C:/VKHCG/03-Hillman/00-RawData/Incoterm_2010.csv
Rows : 1
Columns : 9
############
  Shipping_Term  Seller Carrier Port_From  ... Port_To Terminal Named_Place  Buyer
1           FCA  Seller  Seller     Buyer  ...   Buyer    Buyer       Buyer  Buyer

[1 rows x 9 columns]
### Done!! ############################################
```

(vii) 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='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#########################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#########################################################################
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#########################################################################
### Create the Containers
#########################################################################
containerLength=range(1,21)
containerWidth=range(1,10)
```

```
containerHeigth=range(1,6)
containerStep=1
c=0
for l in containerLength:
 for w in containerWidth:
  for h in containerHeigth:
   containerVolume=(l/containerStep)*(w/containerStep)*(h/containerStep)
c=c+1
ContainerLine=[('ShipType', ['Container']),
('UnitNumber', ('C'+format(c,"06d"))),
('Length',(format(round(l,3),".4f"))),
('Width',(format(round(w,3),".4f"))),
('Height',(format(round(h,3),".4f"))),
('ContainerVolume',(format(round(containerVolume,6),".6f")))]
if c==1:
 ContainerFrame = pd.DataFrame.from_dict(ContainerLine)
else:
 ContainerRow = pd.DataFrame.from_dict(ContainerLine)
ContainerFrame = ContainerFrame.append(ContainerRow)
ContainerFrame.index.name = 'IDNumber'
print('#################')
print('## Container')
print('#################')
print('Rows :',ContainerFrame.shape[0])
print('Columns :',ContainerFrame.shape[1])
print('#################')
#####################################################################
sFileContainerName=sFileDir + '/' + ContainerFileName
ContainerFrame.to_csv(sFileContainerName, index = False)
#####################################################################
## Create valid Boxes with packing foam
#####################################################################
boxLength=range(1,21)
boxWidth=range(1,21)
boxHeigth=range(1,21)
packThick=range(0,6)
boxStep=10
b=0
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=b+1
BoxLine=[('ShipType', ['Box']),
        ('UnitNumber', ('B'+format(b,"06d"))),
('Length',(format(round(l/10,6),".6f"))),
('Width',(format(round(w/10,6),".6f"))),
```

```
('Height',(format(round(h/10,6),".6f"))),
('Thickness',(format(round(t/5,6),".6f"))),
('BoxVolume',(format(round(boxVolume,9),".9f"))),
('ProductVolume',(format(round(productVolume,9),".9f")))]
if b==1:
 BoxFrame = pd.DataFrame.from_dict(BoxLine)
else:
 BoxRow = pd.DataFrame.from_dict(BoxLine)
BoxFrame = BoxFrame.append(BoxRow)
BoxFrame.index.name = 'IDNumber'
print('##################')
print('## Box')
print('##################')
print('Rows :',BoxFrame.shape[0])
print('Columns :',BoxFrame.shape[1])
print('##################')
#########################################################################
sFileBoxName=sFileDir + '/' + BoxFileName
BoxFrame.to_csv(sFileBoxName, index = False)
#########################################################################
## Create valid Product
#########################################################################
productLength=range(1,21)
productWidth=range(1,21)
productHeigth=range(1,21)
productStep=10
p=0
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=p+1
ProductLine=[('ShipType', ['Product']),
('UnitNumber', ('P'+format(p,"06d"))),
('Length',(format(round(l/10,6),".6f"))),
('Width',(format(round(w/10,6),".6f"))),
('Height',(format(round(h/10,6),".6f"))),
('ProductVolume',(format(round(productVolume,9),".9f")))]
if p==1:
 ProductFrame = pd.DataFrame.from_dict(ProductLine)
else:
 ProductRow = pd.DataFrame.from_dict(ProductLine)
ProductFrame = ProductFrame.append(ProductRow)
BoxFrame.index.name = 'IDNumber'
print('##################')
print('## Product')
print('##################')
print('Rows :',ProductFrame.shape[0])
print('Columns :',ProductFrame.shape[1])
```

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

Output:

```
==== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Container-Plan.py ====
################################
Working Base : C:/VKHCG  using  win32
################################
#################
## Container
#################
Rows : 5400
Columns : 2
#################
#################
## Box
#################
Rows : 275880
Columns : 2
#################
#################
## Product
#################
Rows : 48000
Columns : 2
#################
### Done!! ############################################
```

(viii) Create a Delivery Route.

Code:

```
import os
import sys
import pandas as pd
from geopy.distance import vincenty
#########################################################################
InputFileName='GB_Postcode_Warehouse.csv'
OutputFileName='Retrieve_GB_Warehouse.csv'
Company='03-Hillman'
#########################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#########################################################################
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#########################################################################
```

```
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('############')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False)
WarehouseClean=Warehouse[Warehouse.latitude != 0]
WarehouseGood=WarehouseClean[WarehouseClean.longitude != 0]
WarehouseGood.drop_duplicates(subset='postcode', keep='first', inplace=True)
WarehouseGood.sort_values(by='postcode', ascending=1)
#########################################################################
sFileName=sFileDir + '/' + OutputFileName
WarehouseGood.to_csv(sFileName, index = False)
#########################################################################
WarehouseLoop = WarehouseGood.head(20)
for i in range(0,WarehouseLoop.shape[0]):
 print('Run :',i,' ========>>>>>>>>>>>',WarehouseLoop['postcode'][i])
WarehouseHold = WarehouseGood.head(10000)
WarehouseHold['Transaction']=WarehouseHold.apply(lambda row:
'WH-to-WH'
,axis=1)
OutputLoopName='Retrieve_Route_' + 'WH-' + WarehouseLoop['postcode'][i] + '_Route.csv'
WarehouseHold['Seller']=WarehouseHold.apply(lambda row:
'WH-' + WarehouseLoop['postcode'][i]
,axis=1)
```

Output:

```
====== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Route-Plan.py ======
##############################
Working Base : C:/VKHCG  using  win32
##############################
###########
Loading : C:/VKHCG/03-Hillman/00-RawData/GB_Postcode_Warehouse.csv
Run : 0 ========>>>>>>>>>>> AB10
Run : 1 ========>>>>>>>>>>> AB11
Run : 2 ========>>>>>>>>>>> AB12
Run : 3 ========>>>>>>>>>>> AB13
Run : 4 ========>>>>>>>>>>> AB14
Run : 5 ========>>>>>>>>>>> AB15
Run : 6 ========>>>>>>>>>>> AB16
Run : 7 ========>>>>>>>>>>> AB21
Run : 8 ========>>>>>>>>>>> AB22
Run : 9 ========>>>>>>>>>>> AB23
Run : 10 ========>>>>>>>>>>> AB24
Run : 11 ========>>>>>>>>>>> AB25
```

```
Run : 12 ========>>>>>>>>>>> AB30
Run : 13 ========>>>>>>>>>>> AB31
Run : 14 ========>>>>>>>>>>> AB32
Run : 15 ========>>>>>>>>>>> AB33
Run : 16 ========>>>>>>>>>>> AB34
Run : 17 ========>>>>>>>>>>> AB35
Run : 18 ========>>>>>>>>>>> AB36
Run : 19 ========>>>>>>>>>>> AB37
### Done!! ############################################
```

(ix) Person Base Data.

Code:

```
import sys
import os
import shutil
import zipfile
import pandas as pd
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#####################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#####################################################################
Company='04-Clark'
ZIPFiles=['Data_female-names','Data_male-names','Data_last-names']
for ZIPFile in ZIPFiles:
 InputZIPFile=Base+'/'+Company+'/00-RawData/' + ZIPFile + '.zip'
OutputDir=Base+'/'+Company+'/01-Retrieve/01-EDS/02-Python/' + ZIPFile
OutputFile=Base+'/'+Company+'/01-Retrieve/01-EDS/02-Python/Retrieve-'+ZIPFile+'.csv'
zip_file = zipfile.ZipFile(InputZIPFile, 'r')
zip_file.extractall(OutputDir)
zip_file.close()
t=0
for dirname, dirnames, filenames in os.walk(OutputDir):
 for filename in filenames:
  sCSVFile = dirname + '/' + filename
t=t+1
if t==1:
    NameRawData=pd.read_csv(sCSVFile,header=None,low_memory=False)
NameData=NameRawData
 else:
NameRawData=pd.read_csv(sCSVFile,header=None,low_memory=False)
NameData=NameData.append(NameRawData)
NameData.rename(columns={0 : 'NameValues'},inplace=True)
NameData.to_csv(OutputFile, index = False)
shutil.rmtree(OutputDir)
print('Process: ',InputZIPFile)
#####################################################################
print('### Done!! ################################################')
#####################################################################
```

Output:

```
################################
Working Base : C:/VKHCG  using  win32
################################
Process:  C:/VKHCG/04-Clark/00-RawData/Data_female-names.zip
Process:  C:/VKHCG/04-Clark/00-RawData/Data_male-names.zip
Process:  C:/VKHCG/04-Clark/00-RawData/Data_last-names.zip
### Done!! ############################################
```

(x) Connecting to other Data Sources.

Code:

Open MySql
Create a database “DataScience”
Create a python file and add the following code:

```
################# Connection With MySQL ######################
import mysql.connector
conn = mysql.connector.connect(host='localhost',
database='DataScience',

user='root',
password='root')
conn.connect
if(conn.is_connected):
print('###### Connection With MySql Established Successfullly ##### ')
else:
print('Not Connected -- Check Connection Properites')
```

Output:

```
= RESTART: C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/DATA-SCIENCE-PRACTICAL/prac(4b)mysql.py
###### Connection With MySql Established Successfullly #####
>>>
```

PRACTICAL 5

AIM :- ASSESSING DATA

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

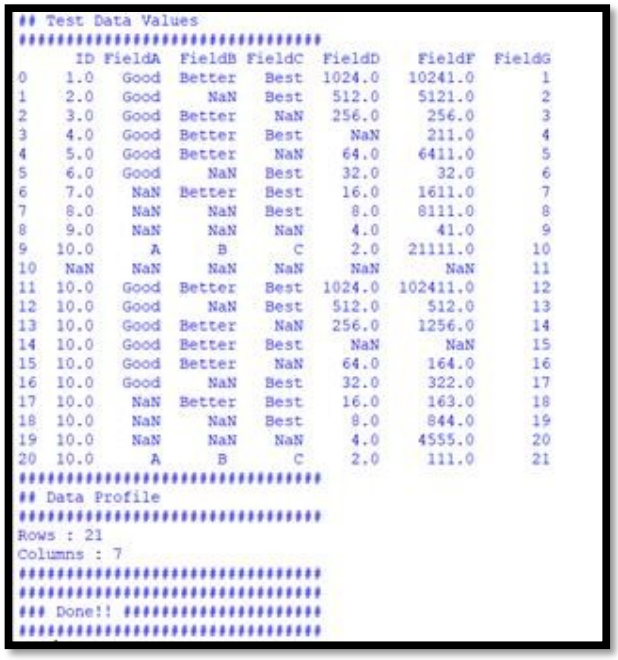
(i) Drop the Columns Where All Elements Are Missing Values

Code:

```
import sys
import os
import pandas as pd
#################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#################################################################
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-01.csv'
Company='01-Vermeulen'
#################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#################################################################
### Import Warehouse
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('##################################')
print('## Raw Data Values')
print('##################################')
print(RawData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('##################################')
#################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
#################################################################
TestData=RawData.dropna(axis=1, how='all')
#################################################################
print('##################################')
```

```
print('## Test Data Values')
print('###################################')
print(TestData)
print('###################################')
print('## Data Profile')
print('###################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('###################################')
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#####################################################################
print('###################################')
print('### Done!! ######################')
print('###################################')
#####################################################################
```

Output:



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

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

Code:

```
import sys
import os
import pandas as pd
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-02.csv'
Company='01-Vermeulen'
#####################################################################
```

```
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#################################################################
### Import Warehouse
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('#################################')
print('## Raw Data Values')
print('#################################')
print(RawData)
print('#################################')
print('## Data Profile')
print('#################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('#################################')
#################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
#################################################################
TestData=RawData.dropna(axis=1, how='any')
#################################################################
print('#################################')
print('## Test Data Values')
print('#################################')
print(TestData)
print('#################################')
print('## Data Profile')
print('#################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('#################################')
#################################################################
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#################################################################
print('#################################')
print('### Done!! ####################')
print('#################################')
#################################################################
```

Output:

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

```
## 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!! ####################
##################################
```

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

Code:

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

```
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-03.csv'
Company='01-Vermeulen'
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
################################################################
print('##################################')
print('Working Base :',Base, ' using Windows ~~~~')
print('##################################')
################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
################################################################
### Import Warehouse
################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('##################################')
print('## Raw Data Values')
print('##################################')
print(RawData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('##################################')
################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
################################################################
TestData=RawData.dropna(thresh=2)
print('##################################')
print('## Test Data Values')
print('##################################')
print(TestData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('##################################')
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
################################################################
print('##################################')
print('### Done!! ####################')
print('##################################')
################################################################
```

Output :-

```
## 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!! ####################
################################
```

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

Code:

```
import sys
import os
import pandas as pd
#######################################################################
pd.options.mode.chained_assignment = None
#######################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#######################################################################
print('##################################')
print('Working Base :',Base, ' using Windows')
print('##################################')
#######################################################################
sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sInputFileName3='01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv'
#######################################################################
sOutputFileName='Assess-Network-Routing-Company.csv'
Company='01-Vermeulen'
#######################################################################
#######################################################################
### Import Country Data
#######################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('##################################')
```

```
print('Loading :',sFileName)
print('###################################')
CountryData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
print('###################################')
#########################################################################
## Assess Country Data
#########################################################################
print('###################################')
print('Changed :',CountryData.columns.values)
CountryData.rename(columns={'Country': 'Country_Name'}, inplace=True)
CountryData.rename(columns={'ISO-2-CODE': 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inplace=True)
CountryData.drop('ISO-3-Code', axis=1, inplace=True)
CountryData.drop('RowID', axis=1, inplace=True)
print('To :',CountryData.columns.values)
print('###################################')
#########################################################################
#########################################################################
### Import Company Data
#########################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('###################################')
print('Loading :',sFileName)
print('###################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('###################################')
#########################################################################
## Assess Company Data
#########################################################################
print('###################################')
print('Changed :',CompanyData.columns.values)
CompanyData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CompanyData.columns.values)
print('###################################')
#########################################################################
#########################################################################
### Import Customer Data
#########################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName3
print('###################################')
print('Loading :',sFileName)
print('###################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
print('###################################')
print('Loaded Customer :',CustomerRawData.columns.values)
print('###################################')
#########################################################################
```

```
CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('#################################')
print('Remove Blank Country Code')
print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])
print('#################################')
#########################################################################
print('#################################')
print('Changed :',CustomerData.columns.values)
CustomerData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CustomerData.columns.values)
print('#################################')
#########################################################################
print('#################################')
print('Merge Company and Country Data')
print('#################################')
CompanyNetworkData=pd.merge(
   CompanyData,
CountryData,
how='inner',
on='Country_Code'
)
#########################################################################
print('#################################')
print('Change ',CompanyNetworkData.columns.values)
for i in CompanyNetworkData.columns.values:
 j='Company_'+i
CompanyNetworkData.rename(columns={i: j}, inplace=True)
print('To ', CompanyNetworkData.columns.values)
print('#################################')
#########################################################################
#########################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#########################################################################
sFileName=sFileDir + '/' + sOutputFileName
print('#################################')
print('Storing :', sFileName)
print('#################################')
CompanyNetworkData.to_csv(sFileName, index = False, encoding="latin-1")
#########################################################################
#########################################################################
print('#################################')
print('### Done!! ######################')
print('#################################')
#########################################################################
```

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

Next, Access the customers location using network router location.

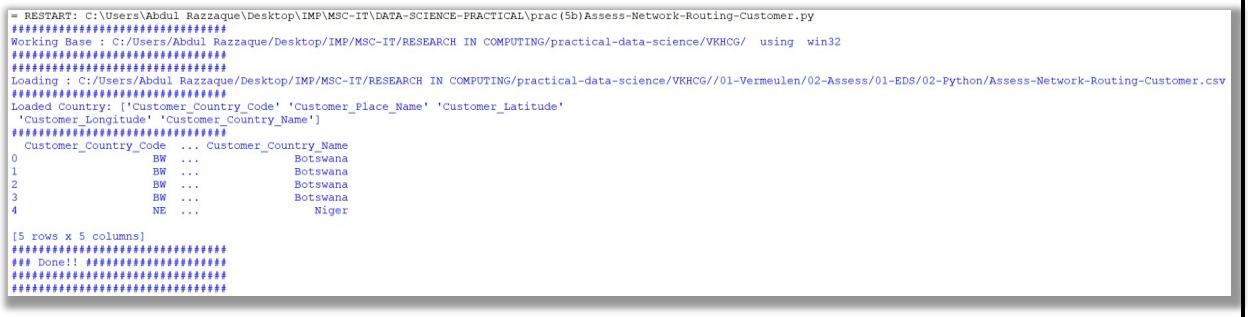
Code:

```
import sys
import os
import pandas as pd
#######################################################################
pd.options.mode.chained_assignment = None
#######################################################################
COMPUTING/practical-data-science/VKHCG/'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#######################################################################

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('#################################')
print('Loading :',sFileName)
print('#################################')
CustomerData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CustomerData.columns.values)
print('#################################')
print(CustomerData.head())
print('#################################')
print('### Done!! ####################')
print('#################################')
#######################################################################
print('#################################')
for n1 in G1.nodes():
 for n2 in G1.nodes():
 if n1 != n2:
   print('Link :',n1,' to ', n2)
G1.add_edge(n1,n2)
```

```
print('#################################')
print('#################################')
print("Nodes of graph: ")
print(G1.nodes())
print("Edges of graph: ")
print(G1.edges())
print('#################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName1
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.draw(G1,pos=nx.spectral_layout(G1),
nodecolor='r',edge_color='g',
with_labels=True,node_size=8000,
font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
print('#################################')
for n1 in G2.nodes():
 for n2 in G2.nodes():
  if n1 != n2:
    print('Link :',n1,' to ', n2)
G2.add_edge(n1,n2)
print('#################################')
print('#################################')
print("Nodes of graph: ")
print(G2.nodes())
print("Edges of graph: ")
print(G2.edges())
print('#################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName2
print('#################################')
print('Storing :', sFileName)
print('#################################')
nx.draw(G2,pos=nx.spectral_layout(G2),
nodecolor='r',edge_color='b',
with_labels=True,node_size=8000,
font_size=12)
plt.savefig(sFileName) # save as png
```

```
plt.show() # display
#########################################################################
```

Output:



```
= RESTART: C:\Users\Abdul Razzaque\Desktop\IMP\MSC-IT\DATA-SCIENCE-PRACTICAL\prac(5b)Assess-Network-Routing-Customer.py
################################
Working Base : C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/RESEARCH IN COMPUTING/practical-data-science/VKHCG/  using  win32
################################
################################
Loading : C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/RESEARCH IN COMPUTING/practical-data-science/VKHCG//01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
################################
  Customer_Country_Code  ... Customer_Country_Name
0                    BW  ...              Botswana
1                    BW  ...              Botswana
2                    BW  ...              Botswana
3                    BW  ...              Botswana
4                    NE  ...                 Niger

[5 rows x 5 columns]
################################
### Done!! ####################
################################
################################
```

Assess-Network-Routing-Node.py

Code:

```
import sys
import os
import pandas as pd
#########################################################################
pd.options.mode.chained_assignment = None
#########################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/RESEARCH IN
COMPUTING/practical-data-science/VKHCG/'
#########################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#########################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
#########################################################################
sOutputFileName='Assess-Network-Routing-Node.csv'
Company='01-Vermeulen'
#########################################################################
### Import IP Data
#########################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
IPData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded IP :', IPData.columns.values)
print('##################################')
#########################################################################
print('##################################')
print('Changed :',IPData.columns.values)
IPData.drop('RowID', axis=1, inplace=True)
IPData.drop('ID', axis=1, inplace=True)
IPData.rename(columns={'Country': 'Country_Code'}, inplace=True)
IPData.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
IPData.rename(columns={'Post.Code': 'Post_Code'}, inplace=True)
```

```
IPData.rename(columns={'First.IP.Number': 'First_IP_Number'}, inplace=True)
IPData.rename(columns={'Last.IP.Number': 'Last_IP_Number'}, inplace=True)
print('To :',IPData.columns.values)
print('##################################')
######################################################################
print('##################################')
print('Change ',IPData.columns.values)
for i in IPData.columns.values:
 j='Node_'+i
IPData.rename(columns={i: j}, inplace=True)
print('To ', IPData.columns.values)
print('##################################')
```

Output:

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

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

Code:

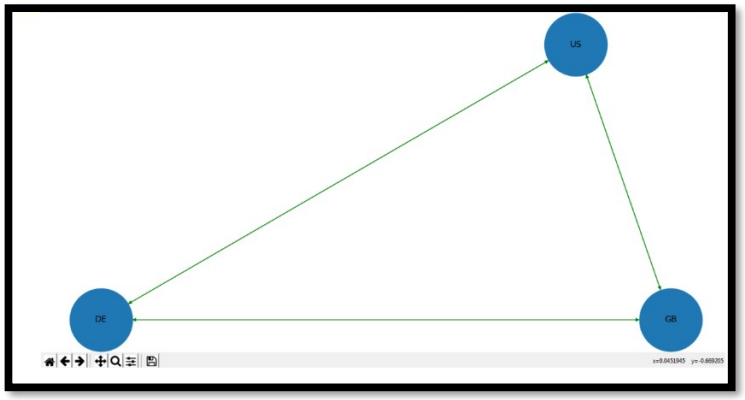
```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
######################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/RESEARCH IN
COMPUTING/practical-data-science/VKHCG'
######################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
```

```
#####################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName1='Assess-DAG-Company-Country.png'
sOutputFileName2='Assess-DAG-Company-Country-Place.png'
Company='01-Vermeulen'
#####################################################################
### Import Company Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('#################################')
#####################################################################
print(CompanyData)
print('#################################')
print('Rows : ',CompanyData.shape[0])
print('#################################')
#####################################################################
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)
print('#################################')
for n1 in G1.nodes():
 for n2 in G1.nodes():
  if n1 != n2:
   print('Link :',n1,' to ', n2)
G1.add_edge(n1,n2)
print('#################################')
print('#################################')
print("Nodes of graph: ")
print(G1.nodes())
print("Edges of graph: ")
print(G1.edges())
print('#################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName1
print('#################################')
print('Storing :', sFileName)
print('#################################')
```

```
nx.draw(G1,pos=nx.spectral_layout(G1),
nodecolor='r',edge_color='g',
with_labels=True,node_size=8000,
font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
print('###################################')
for n1 in G2.nodes():
 for n2 in G2.nodes():
  if n1 != n2:
   print('Link :',n1,' to ', n2)
G2.add_edge(n1,n2)
print('###################################')
print('###################################')
print("Nodes of graph: ")
print(G2.nodes())
print("Edges of graph: ")
print(G2.edges())
print('###################################')
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#####################################################################
sFileName=sFileDir + '/' + sOutputFileName2
print('###################################')
print('Storing :', sFileName)
print('###################################')
nx.draw(G2,pos=nx.spectral_layout(G2),
       nodecolor='r',edge_color='b',
with_labels=True,node_size=8000,
font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
```

Output:

```
= RESTART: C:\Users\Abdul Razzaque\Desktop\IMP\MSC-IT\DATA-SCIENCE-PRACTICAL\prac(5c)Assess-DAG-Location.py
################################
Working Base : C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/RESEARCH IN COMPUTING/practical-data-science/VKHCG  using  win32
################################
################################
Loading : C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/RESEARCH IN COMPUTING/practical-data-science/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/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:
[('GB', 'GB')]
################################
################################
Storing : C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/RESEARCH IN COMPUTING/practical-data-science/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-DAG-Company-Country.png
################################
```



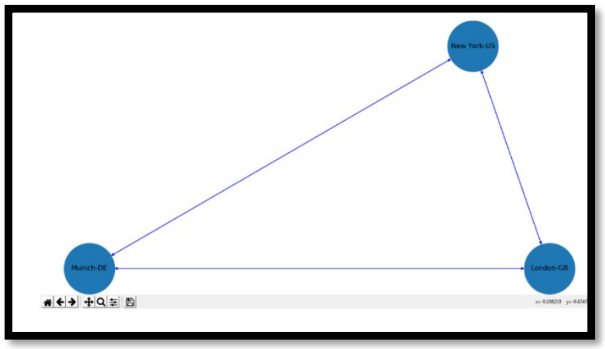
Customer Location DAG

Code:

```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
##################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/RESEARCH IN
COMPUTING/practical-data-science/VKHCG'
##################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
##################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName1='Assess-DAG-Company-Country.png'
sOutputFileName2='Assess-DAG-Company-Country-Place.png'
Company='01-Vermeulen'
##################################################################
```

```
### Import Company Data
#########################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('##################################')
print(CompanyData)
print('##################################')
print('Rows : ',CompanyData.shape[0])
print('##################################')
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)
```

Output:



Open your Python editor and create a file named Assess-DAG-GPS.py in directory C:\VKHCG\01-Vermeulen\02-Assess.

Code:

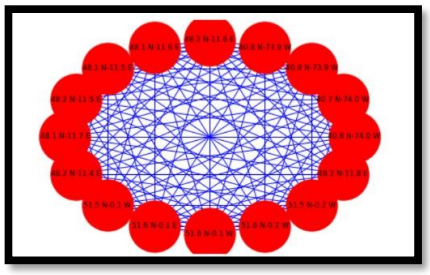
```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName='Assess-DAG-Company-GPS.png'
Company='01-Vermeulen'
```

```
### Import Company Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('################################')
print(CompanyData)
print('################################')
print('Rows : ',CompanyData.shape[0])
print('################################')
G=nx.Graph()
for i in range(CompanyData.shape[0]):
nLatitude=round(CompanyData['Latitude'][i],2)
nLongitude=round(CompanyData['Longitude'][i],2)
if nLatitude < 0:
 sLatitude = str(nLatitude*-1) + ' S'
else:
 sLatitude = str(nLatitude) + ' N'
if nLongitude < 0:
 sLongitude = str(nLongitude*-1) + ' W'
else:
 sLongitude = str(nLongitude) + ' E'
sGPS= sLatitude + '-' + sLongitude
G.add_node(sGPS)
print('################################')
for n1 in G.nodes():
 for n2 in G.nodes():
  if n1 != n2:
   print('Link :',n1,' to ', n2)
G.add_edge(n1,n2)
print('################################')
print('################################')
print("Nodes of graph: ")
print(G.number_of_nodes())
print("Edges of graph: ")
print(G.number_of_edges())
print('################################')
```

Output:

```
################################
Working Base : C:/VKHCG using win32
################################
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-
Python/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
-
-
-
Link : 48.15 N-11.74 E to 48.15 N-11.46 E
Link : 48.15 N-11.74 E to 48.09 N-11.54 E
Link : 48.15 N-11.74 E to 48.18 N-11.75 E
Link : 48.15 N-11.74 E to 48.1 N-11.47 E
##################################
Nodes of graph:
117
Edges of graph:
6786
##################################



D. 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
#####################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
sOutputFileName='Assess-DE-Billboard-Visitor.csv'
Company='02-Krennwallner'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'
```

```
conn = sq.connect(sDatabaseName)
#####################################################################
### Import Billboard Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('#################################')
print('Loading :',sFileName)
print('#################################')
BillboardRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
BillboardRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BillboardData=BillboardRawData
print('Loaded Company :',BillboardData.columns.values)
print('#################################')
#####################################################################
print('################')
sTable='Assess_BillboardData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardData.to_sql(sTable, conn, if_exists="replace")
print('################')
#####################################################################
print(BillboardData.head())
print('#################################')
print('Rows : ',BillboardData.shape[0])
print('#################################')
#####################################################################
### Import Billboard Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('#################################')
print('Loading :',sFileName)
print('#################################')
VisitorRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData[VisitorRawData.Country=='DE']
print('Loaded Company :',VisitorData.columns.values)
print('#################################')
#####################################################################
print('################')
sTable='Assess_VisitorData'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print('################')
#####################################################################
print(VisitorData.head())
print('#################################')
print('Rows : ',VisitorData.shape[0])
print('#################################')
#####################################################################
print('################')
```

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

Output:

C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db Table:

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

E. Write a Python / R program to generate GML file from the given csv file.

Code:

```
import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
Company='02-Krennwallner'
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'
conn = sq.connect(sDatabaseName)
################################################################
### Import Country Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
VisitorRawData=pd.read_csv(sFileName,
header=0,
low_memory=False,
encoding="latin-1",
skip_blank_lines=True)
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData
print('Loaded Company :',VisitorData.columns.values)
print('##################################')
```

```
#####################################################################
print('#################')
sTable='Assess_Visitor'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#####################################################################
print(VisitorData.head())
print('#################################')
print('Rows : ',VisitorData.shape[0])
print('#################################')
#####################################################################
print('#################')
sView='Assess_Visitor_UseIt'
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+ " Place_Name is not null;"
sql.execute(sSQL,conn)
#####################################################################
print('#################')
sView='Assess_Total_Visitors_Location'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " Country,"
sSQL=sSQL+ " Place_Name,"
sSQL=sSQL+ " SUM(UsesIt) AS TotalUsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor_UseIt"
sSQL=sSQL+ " GROUP BY"
sSQL=sSQL+ " Country,"
sSQL=sSQL+ " Place_Name"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " TotalUsesIt DESC"
sSQL=sSQL+ " LIMIT 10;"
```

```
sql.execute(sSQL,conn)
#########################################################################
print('#################')
sView='Assess_Total_Visitors_GPS'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " Latitude,"
sSQL=sSQL+ " Longitude,"
sSQL=sSQL+ " SUM(UsesIt) AS TotalUsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor_UseIt"
sSQL=sSQL+ " GROUP BY"
sSQL=sSQL+ " Latitude,"
sSQL=sSQL+ " Longitude"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " TotalUsesIt DESC"
sSQL=sSQL+ " LIMIT 10;"
sql.execute(sSQL,conn)
#########################################################################
sTables=['Assess_Total_Visitors_Location', 'Assess_Total_Visitors_GPS']
for sTable in sTables:
 print('#################')
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
TopData=pd.read_sql_query(sSQL, conn)
print('#################')
print(TopData)
print('#################')
print('#################################')
print('Rows : ',TopData.shape[0])
print('#################################')
#########################################################################
print('### Done!! ############################################')
#########################################################################
```

Output:

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

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

Code:

```
import os
import pandas as pd
from geopy.geocoders import Nominatim
geolocator = Nominatim()
##################################################################
InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_GB_Postcode_Warehouse.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_GB_Warehouse_Address.csv'
Company='03-Hillman'
##################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using Windows')
print('##################################')
##################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
##################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
##################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('############')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False)
```

```
Warehouse.sort_values(by='postcode', ascending=1)
#####################################################################
## Limited to 10 due to service limit on Address Service.
#####################################################################
WarehouseGoodHead=Warehouse[Warehouse.latitude != 0].head(5)
WarehouseGoodTail=Warehouse[Warehouse.latitude != 0].tail(5)
#####################################################################
WarehouseGoodHead['Warehouse_Point']=WarehouseGoodHead.apply(lambda row:
(str(row['latitude'])+','+str(row['longitude']))
,axis=1)
WarehouseGoodHead['Warehouse_Address']=WarehouseGoodHead.apply(lambda row:
geolocator.reverse(row['Warehouse_Point']).address
,axis=1)
WarehouseGoodHead.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodHead.drop('id', axis=1, inplace=True)
WarehouseGoodHead.drop('postcode', axis=1, inplace=True)
#####################################################################
WarehouseGoodTail['Warehouse_Point']=WarehouseGoodTail.apply(lambda row:
(str(row['latitude'])+','+str(row['longitude']))
,axis=1)
WarehouseGoodTail['Warehouse_Address']=WarehouseGoodTail.apply(lambda row:
geolocator.reverse(row['Warehouse_Point']).address
,axis=1)
WarehouseGoodTail.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodTail.drop('id', axis=1, inplace=True)
WarehouseGoodTail.drop('postcode', axis=1, inplace=True)
#####################################################################
WarehouseGood=WarehouseGoodHead.append(WarehouseGoodTail, ignore_index=True)
print(WarehouseGood)
#####################################################################
sFileName=sFileDir + '/' + OutputFileName
WarehouseGood.to_csv(sFileName, index = False)
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

Output:

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

PRACTICAL 6

AIM :- PROCESSING DATA

Build the time hub, links, and satellites.

Code :-

```
#################################################################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
from datetime import datetime
from datetime import timedelta
from pytz import timezone, all_timezones
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
pd.options.mode.chained_assignment = None
#################################################################
if sys.platform == 'linux':
 Base=os.path.expanduser('~') + '/VKHCG'
else:
 Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#################################################################
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
#################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
#################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
 os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#################################################################
base = datetime(2018,1,1,0,0,0)
numUnits=10*365*24
#################################################################
```

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

```
('DateTimeKey', [DateTimeKey]),
('UTCDateTimeValue', [sDateTime]),
('Zone', [zone]),
('DateTimeValue', [sZoneDateTime])]
if t==1:
 TimeZoneFrame = pd.DataFrame.from_items(TimeZoneLine)
else:
 TimeZoneRow = pd.DataFrame.from_items(TimeZoneLine)
TimeZoneFrame = TimeZoneFrame.append(TimeZoneRow)
TimeZoneFrameIndex=TimeZoneFrame.set_index(['IDZoneNumber'],inplace=False)
sZone=zone.replace('/','-').replace(' ','')
#################################################################
sTable = 'Process-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn1, if_exists="replace")
#################################################################
#################################################################
sTable = 'Satellite-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('#################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#################')
#################################################################
print('### Done!! ############################################')
#################################################################
```

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!! ############################################
```

Golden Nominal

Code :-

```
import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
from datetime import datetime, timedelta
from pytz import timezone, all_timezones
from random import randint
import uuid
#####################################################################
if sys.platform == 'linux':
 Base=os.path.expanduser('~') + '/VKHCG'
else:
 Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
Company='04-Clark'
sInputFileName='02-Assess/01-EDS/02-Python/Assess_People.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
 os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
### Import Female Data
#####################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
print(sFileName)
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)
start_date = datetime(1900,1,1,0,0,0)
start_date_utc=start_date.replace(tzinfo=timezone('UTC'))
HoursBirth=100*365*24
```

```
RawData['BirthDateUTC']=RawData.apply(lambda row:
(start_date_utc + timedelta(hours=randint(0, HoursBirth)))
,axis=1)
zonemax=len(all_timezones)-1
RawData['TimeZone']=RawData.apply(lambda row:
(all_timezones[randint(0, zonemax)])
,axis=1)
RawData['BirthDateISO']=RawData.apply(lambda row:
row["BirthDateUTC"].astimezone(timezone(row['TimeZone']))
,axis=1)
RawData['BirthDateKey']=RawData.apply(lambda row:
row["BirthDateUTC"].strftime("%Y-%m-%d %H:%M:%S")
,axis=1)
RawData['BirthDate']=RawData.apply(lambda row:
row["BirthDateISO"].strftime("%Y-%m-%d %H:%M:%S")
,axis=1)
RawData['PersonID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1)
#####################################################################
Data=RawData.copy()
Data.drop('BirthDateUTC', axis=1,inplace=True)
Data.drop('BirthDateISO', axis=1,inplace=True)
indexed_data = Data.set_index(['PersonID'])
print('##################################')
#####################################################################
print('#################')
sTable='Process_Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_data.to_sql(sTable, conn1, if_exists="replace")
print('#################')
#####################################################################
PersonHubRaw=Data[['PersonID','FirstName','SecondName','LastName','BirthDateKey']]
PersonHubRaw['PersonHubID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1)
PersonHub=PersonHubRaw.drop_duplicates(subset=None, \
keep='first',\
inplace=False)

indexed_PersonHub = PersonHub.set_index(['PersonHubID'])
sTable = 'Hub-Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonHub.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
PersonSatelliteGenderRaw=Data[['PersonID','FirstName','SecondName','LastName'\
,'BirthDateKey','Gender']]
PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1)
```

```
PersonSatelliteGender=PersonSatelliteGenderRaw.drop_duplicates(subset=None, \
keep='first', \
inplace=False)

indexed_PersonSatelliteGender = PersonSatelliteGender.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Gender'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteGender.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
PersonSatelliteBirthdayRaw=Data[['PersonID','FirstName','SecondName','LastName',\
'BirthDateKey','TimeZone','BirthDate']]
PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lambda row:
str(uuid.uuid4())
,axis=1)
PersonSatelliteBirthday=PersonSatelliteBirthdayRaw.drop_duplicates(subset=None, \
keep='first',\
inplace=False)
indexed_PersonSatelliteBirthday = PersonSatelliteBirthday.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Names'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteBirthday.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
sFileDir=Base + '/' + Company + '/03-Process/01-EDS/02-Python'
if not os.path.exists(sFileDir):
 os.makedirs(sFileDir)
#####################################################################
sOutputFileName = sTable + '.csv'
sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
#####################################################################
print('################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('################')
#####################################################################
print('### Done!! ############################################')
#####################################################################
```

Output :-

```
==== RESTART: C:\VKHCG\03-Hillman\03-Process\Process-Vehicle-Logistics.py ====
##################################
go forward, hold to see history
Working Base : C:/VKHCG  using  win32
################################
###########
Loading : C:/VKHCG/03-Hillman/00-RawData/VehicleData.csv
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process_Vehicles
Storing : C:/VKHCG/88-DV/datavault.db  Table: Hub-Object-Vehicle
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Object-Make-Model
Storing : C:/VKHCG/88-DV/datavault.db  View: Dim-Object
################
Loading : C:/VKHCG/88-DV/datavault.db  Table: Dim-Object
################
                VehicleMake                       VehicleModel
ObjectDimID
2213             AM General                 DJ Po Vehicle 2WD
2212             AM General                   FJ8c Post Office
129              AM General               Post Office DJ5 2WD
131              AM General               Post Office DJ8 2WD
2869       ASC Incorporated                               GNX
...                     ...                               ...
1996                 smart                fortwo convertible
1997                 smart                      fortwo coupe
2622                 smart    fortwo electric drive cabriolet
2833                 smart  fortwo electric drive convertible
2623                 smart        fortwo electric drive coupe

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

Vehicles

Code :-

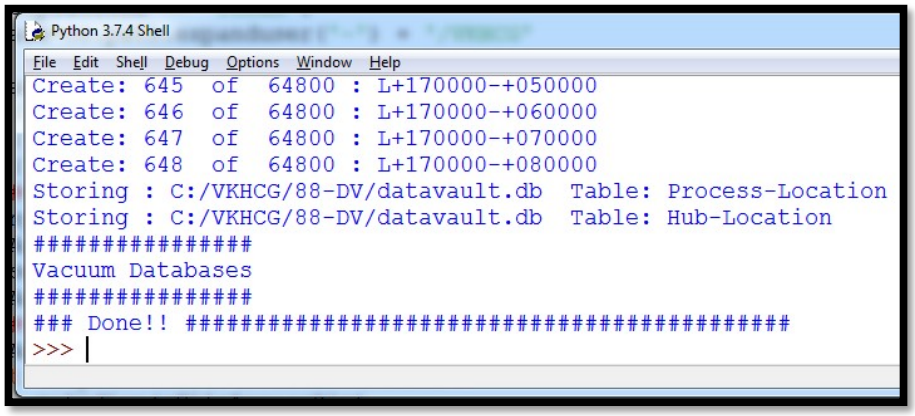
```
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
pd.options.mode.chained_assignment = None
################################################################
if sys.platform == 'linux':
 Base=os.path.expanduser('~') + '/VKHCG'
else:
 Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG/'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
Company='03-Hillman'
```

```
InputDir='00-RawData'
InputFileName='VehicleData.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
 os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('############')
print('Loading :',sFileName)
VehicleRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#####################################################################
sTable='Process_Vehicles'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleRaw.to_sql(sTable, conn1, if_exists="replace")
#####################################################################
VehicleRawKey=VehicleRaw[['Make','Model']].copy()
VehicleKey=VehicleRawKey.drop_duplicates()
#####################################################################
VehicleKey['ObjectKey']=VehicleKey.apply(lambda row:
str('('+ str(row['Make']).strip().replace(' ', '-').replace('/', '-').lower() +
')-(' + (str(row['Model']).strip().replace(' ', '-').replace(' ', '-').lower())
+')')
,axis=1)
#####################################################################
VehicleKey['ObjectType']=VehicleKey.apply(lambda row:
'vehicle'
,axis=1)
#####################################################################
VehicleKey['ObjectUUID']=VehicleKey.apply(lambda row:
str(uuid.uuid4())
,axis=1)
#####################################################################
### Vehicle Hub
#####################################################################
#
VehicleHub=VehicleKey[['ObjectType','ObjectKey','ObjectUUID']].copy()
VehicleHub.index.name='ObjectHubID'
sTable = 'Hub-Object-Vehicle'
print('Storing :',sDatabaseName,' Table:',sTable)
```

```
VehicleHub.to_sql(sTable, conn2, if_exists="replace")
#########################################################################
### Vehicle Satellite
#########################################################################
#
VehicleSatellite=VehicleKey[['ObjectType','ObjectKey','ObjectUUID','Make','Model']].copy(
)
VehicleSatellite.index.name='ObjectSatelliteID'
sTable = 'Satellite-Object-Make-Model'
print('Storing :',sDatabaseName,' Table:',sTable)
VehicleSatellite.to_sql(sTable, conn2, if_exists="replace")

#########################################################################
### Vehicle Dimension
#########################################################################
sView='Dim-Object'
print('Storing :',sDatabaseName,' View:',sView)
sSQL="CREATE VIEW IF NOT EXISTS [" + sView + "] AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " H.ObjectType,"
sSQL=sSQL+ " H.ObjectKey AS VehicleKey,"
sSQL=sSQL+ " TRIM(S.Make) AS VehicleMake,"
sSQL=sSQL+ " TRIM(S.Model) AS VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " [Hub-Object-Vehicle] AS H"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " [Satellite-Object-Make-Model] AS S"
sSQL=sSQL+ " ON"
sSQL=sSQL+ " H.ObjectType=S.ObjectType"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " H.ObjectUUID=S.ObjectUUID;"
sql.execute(sSQL,conn2)
print('#################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " VehicleMake,"
sSQL=sSQL+ " VehicleModel"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " [" + sView + "]"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " VehicleMake"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " VehicleMake;"
DimObjectData=pd.read_sql_query(sSQL, conn2)
DimObjectData.index.name='ObjectDimID'
DimObjectData.sort_values(['VehicleMake','VehicleModel'],inplace=True, ascending=True)
print('#################')
print(DimObjectData)
#########################################################################
print('#################')
```

```
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#################')
#########################################################################
conn1.close()
conn2.close()
#########################################################################
#print('### Done!! ##############################################')
#########################################################################
```

Output :-



Human-Environment Interaction

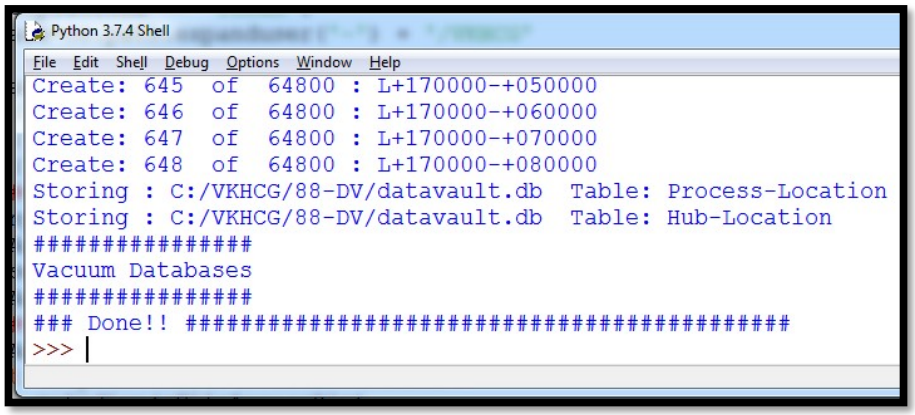
Code :-

```
#########################################################################
# -*- coding: utf-8 -*-
#########################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG/'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#########################################################################
Company='01-Vermeulen'
InputAssessGraphName='Assess_All_Animals.gml'
EDSAssessDir='02-Assess/01-EDS'
InputAssessDir=EDSAssessDir + '/02-Python'
#########################################################################
```

```
sFileAssessDir=Base + '/' + Company + '/' + InputAssessDir
if not os.path.exists(sFileAssessDir):
os.makedirs(sFileAssessDir)
########################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
########################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
########################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
 os.makedirs(sDataBaseDir)
########################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
t=0
tMax=360*180
########################################################################
for Longitude in range(-180,180,10):
 for Latitude in range(-90,90,10):
  t+=1
IDNumber=str(uuid.uuid4())
LocationName='L'+format(round(Longitude,3)*1000, '+07d') +\
'-'+format(round(Latitude,3)*1000, '+07d')
print('Create:',t,' of ',tMax,':',LocationName)
LocationLine=[('ObjectBaseKey', ['GPS']),
('IDNumber', [IDNumber]),
('LocationNumber', [str(t)]),
('LocationName', [LocationName]),
('Longitude', [Longitude]),
('Latitude', [Latitude])]
if t==1:
 LocationFrame = pd.DataFrame.from_items(LocationLine)
else:
 LocationRow = pd.DataFrame.from_items(LocationLine)
LocationFrame = LocationFrame.append(LocationRow)
########################################################################
LocationHubIndex=LocationFrame.set_index(['IDNumber'],inplace=False)
########################################################################
sTable = 'Process-Location'
print('Storing :',sDatabaseName,' Table:',sTable)
LocationHubIndex.to_sql(sTable, conn1, if_exists="replace")
########################################################################
sTable = 'Hub-Location'
print('Storing :',sDatabaseName,' Table:',sTable)
LocationHubIndex.to_sql(sTable, conn2, if_exists="replace")
########################################################################
print('################')
```

```
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#################')
########################################################################
print('### Done!! ################################################')
########################################################################
```

Output :-



Forecasting

Code :-

```
########################################################################
import sys
import os
import sqlite3 as sq
import quandl
import pandas as pd
########################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG/'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
########################################################################
Company='04-Clark'
sInputFileName='00-RawData/VKHCG_Shares.csv'
sOutputFileName='Shares.csv'
########################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
########################################################################
sFileDir1=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir1):
```

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

```
 sShareName=str(RawData['Shares'][sShare])
ShareData = quandl.get(sShareName)
UnitsOwn=RawData['Units'][sShare]
ShareData['UnitsOwn']=ShareData.apply(lambda row:(UnitsOwn),axis=1)
ShareData['ShareCode']=ShareData.apply(lambda row:(sShareName),axis=1)
print('#################')
print('Share :',sShareName)
print('Rows :',ShareData.shape[0])
print('Columns:',ShareData.shape[1])
print('#################')
#########################################################################
print('#################')
sTable=str(RawData['sTable'][sShare])
print('Storing :',sDatabaseName,' Table:',sTable)
ShareData.to_sql(sTable, conn, if_exists="replace")
print('#################')
#########################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
print('#################################')
print('Storing :', sFileName)
print('#################################')
ShareData.to_csv(sFileName, index = False)
print('#################################')
#########################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir2 + '/Assess_' + sOutputFileName
print('#################################')
print('Storing :', sFileName)
print('#################################')
ShareData.to_csv(sFileName, index = False)
print('#################################')
#########################################################################
sOutputFileName = sTable.replace("/","-") + '.csv'
sFileName=sFileDir3 + '/Process_' + sOutputFileName
print('#################################')
print('Storing :', sFileName)
print('#################################')
ShareData.to_csv(sFileName, index = False)
print('#################################')
print('### Done!! ############################################')
#########################################################################
```

Output :-

```
========== RESTART: C:\VKHCG\04-Clark\03-Process\Process-Shares-Data.py ==========
Working Base : C:/VKHCG using win32
Loading : C:/VKHCG/04-Clark/00-RawData/VKHCG  Shares.csv
Rows : 10
Columns: 3
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve  Shares.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_Shares.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process  Shares.csv
Share : WIKI/GOOGL
Rows : 3424
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Google
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Google.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Google.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Google.csv
Share : WIKI/MSFT
Rows : 8076
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Microsoft
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Microsoft.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Microsoft.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Microsoft.csv
Share : WIKI/UPS
Rows : 4622
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_UPS
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_UPS.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_UPS.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_UPS.csv
Share : WIKI/AMZN
Rows : 5248
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: WIKI_Amazon
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Amazon.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Amazon.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_WIKI_Amazon.csv
Share : LOCALBTC/USD
Rows : 1863
Columns: 6
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db Table: LOCALBTC_USD
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_LOCALBTC_USD.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_LOCALBTC_USD.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_LOCALBTC_USD.csv
Share : PERTH/AUD_USD_M
Rows : 340 ### Done!! ############################################
```

PRACTICAL 7

AIM :- TRANSFORMING DATA

Transform Superstep

Code :-

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG/'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#####################################################################
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
 os.makedirs(sDataVaultDir)
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
 os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
#####################################################################
```

```
print('\n###################################')
print('Time Category')
print('UTC Time')
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
print(BirthDateZoneUTCStr)
print('###################################')
print('Birth Date in Reykjavik :')
BirthZone = 'Atlantic/Reykjavik'
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
print(BirthDateStr)
print('###################################')
#####################################################################
IDZoneNumber=str(uuid.uuid4())
sDateTimeKey=BirthDateZoneStr.replace(' ','-').replace(':','-')
TimeLine=[('ZoneBaseKey', ['UTC']),
('IDNumber', [IDZoneNumber]),
('DateTimeKey', [sDateTimeKey]),
('UTCDateTimeValue', [BirthDateZoneUTC]),
('Zone', [BirthZone]),
('DateTimeValue', [BirthDateStr])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
#####################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Hub-Time-Gunnarsson'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Time-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']]
TimeSatelliteIndex=TimeSatellite.set_index(['IDNumber'],inplace=False)
#####################################################################
BirthZoneFix=BirthZone.replace(' ','-').replace('/','-')
sTable = 'Satellite-Time-' + BirthZoneFix + '-Gunnarsson'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
TimeSatelliteIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Time-' + BirthZoneFix + '-Gunnarsson'
TimeSatelliteIndex.to_sql(sTable, conn3, if_exists="replace")
```

```
#####################################################################
print('\n###################################')
print('Person Category')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
print('Name:',FirstName,LastName)
print('Birth Date:',BirthDateLocal)
print('Birth Zone:',BirthZone)
print('UTC Birth Date:',BirthDateZoneStr)
print('###################################')
#####################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('IDNumber', [IDPersonNumber]),
('FirstName', [FirstName]),
('LastName', [LastName]),
('Zone', ['UTC']),
('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
#####################################################################
TimeHub=PersonFrame
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Hub-Person-Gunnarsson'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Person-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
```

Output :-

```
>>>
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson_is_Born.py
Working Base : C:/VKHCG  using  win32
Time Category
UTC Time
1960-12-20 10:15:00 (UTC) (+0000)
###################################
Birth Date in Reykjavik :
1960-12-20 09:15:00 (-01) (-0100)
###################################
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Time-Gunnarsson
###################################
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Satellite-Time-Atlantic-Reykjavik-Gunnarsson
###################################
Person Category
Name: Guðmundur Gunnarsson
Birth Date: 1960-12-20 09:15:00
Birth Zone: Atlantic/Reykjavik
UTC Birth Date: 1960-12-20 10:15:00
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Person-Gunnarsson
###################################
```

You must build three items: dimension Person, dimension Time, and factPersonBornAtTime.

Code :-

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
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:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG/'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
 os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir):
os.makedirs(sDataWarehousetDir)
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
print('\n#################################')
print('Time Dimension')
BirthZone = 'Atlantic/Reykjavik'
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
#####################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [IDTimeNumber]),
('UTCDate', [BirthDateZoneStr]),
('LocalTime', [BirthDateLocal]),
('TimeZone', [BirthZone])]
```

```
TimeFrame = pd.DataFrame.from_items(TimeLine)
#####################################################################
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
#####################################################################
sTable = 'Dim-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2,
if_exists="replace")print('\n#################################')
print('Dimension Person')
print('\n#################################')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [IDPersonNumber]),
('FirstName', [FirstName]),
('LastName', [LastName]),
('Zone', ['UTC']),
('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
#####################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####################################################################
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
print('\n#################################')
print('Fact - Person - time')
print('\n#################################')
IDFactNumber=str(uuid.uuid4())
PersonTimeLine=[('IDNumber', [IDFactNumber]),
('IDPersonNumber', [IDPersonNumber]),
('IDTimeNumber', [IDTimeNumber])]
PersonTimeFrame = pd.DataFrame.from_items(PersonTimeLine)
#####################################################################
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
#####################################################################
sTable = 'Fact-Person-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
```

Output :-

```
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson-Sun-Model.py
################################
Working Base : C:/VKHCG  using  win32
################################

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

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

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

################################
```

Open the Transform-Sun-Models.py file

Code :-

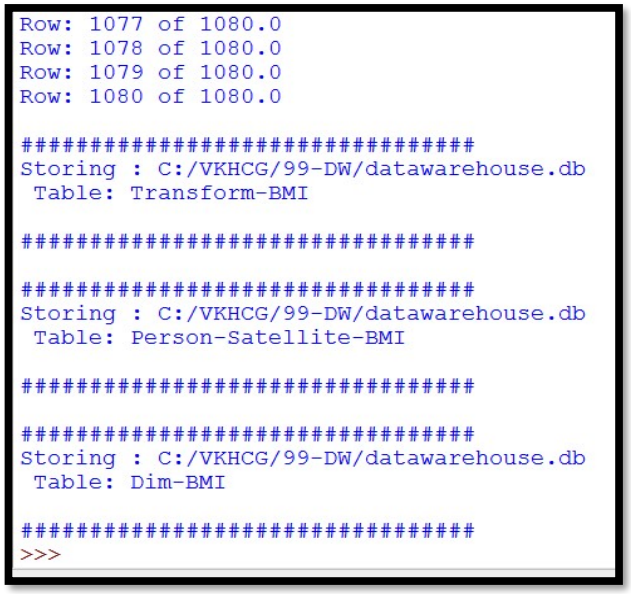
```
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
####################################################################
if sys.platform == 'linux':
 Base=os.path.expanduser('~') + '/VKHCG'
else:
 Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG/'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
####################################################################
Company='01-Vermeulen'
####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
####################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
 os.makedirs(sDataVaultDir)
```

```
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
 os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
#####################################################################
sSQL=" SELECT DateTimeValue FROM [Hub-Time];"
DateDataRaw=pd.read_sql_query(sSQL, conn2)
DateData=DateDataRaw.head(1000)
print(DateData)
#####################################################################
print('\n#################################')
print('Time Dimension')
print('\n#################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
 BirthZone = ('Atlantic/Reykjavik','Europe/London','UCT')
for j in range(len(BirthZone)):
 t+=1
print(t,mt*3)
BirthDateUTC = datetime.strptime(DateData['DateTimeValue'][i],"%Y-%m-%d
%H:%M:%S")
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone[j]))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
#####################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [str(IDTimeNumber)]),
('UTCDate', [str(BirthDateZoneStr)]),
('LocalTime', [str(BirthDateLocal)]),
('TimeZone', [str(BirthZone)])]
if t==1:
 TimeFrame = pd.DataFrame.from_items(TimeLine)
else:
 TimeRow = pd.DataFrame.from_items(TimeLine)
TimeFrame=TimeFrame.append(TimeRow)
#####################################################################
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
#####################################################################
```

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

```
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
```

Output :-



Simple Linear Regression

Code :-

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG/'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
```

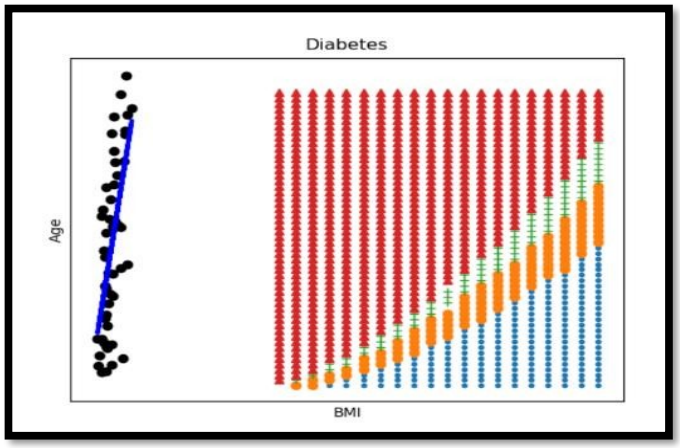
```
 os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
 os.makedirs(sDataVaultDir)
#####################################################################
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
 os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
#####################################################################
t=0
tMax=((300-100)/10)*((300-30)/5)
for heightSelect in range(100,300,10):
 for weightSelect in range(30,300,5):
  height = round(heightSelect/100,3)
weight = int(weightSelect)
bmi = weight/(height*height)
if bmi <= 18.5:
 BMI_Result=1
elif bmi > 18.5 and bmi < 25:
 BMI_Result=2
elif bmi > 25 and bmi < 30:
 BMI_Result=3
elif bmi > 30:
 BMI_Result=4
else:
 BMI_Result=0
PersonLine=[('PersonID', [str(t)]),
           ('Height', [height]),
('Weight', [weight]),
('bmi', [bmi]),
('Indicator', [BMI_Result])]
t+=1
print('Row:',t,'of',tMax)
if t==1:
 PersonFrame = pd.DataFrame.from_items(PersonLine)
else:
 PersonRow =  pd.DataFrame.from_items(PersonLine)
PersonFrame = PersonFrame.append(PersonRow)
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
```

```
#####################################################################
sTable = 'Transform-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
#####################################################################
#####################################################################
sTable = 'Person-Satellite-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
fig = plt.figure()
PlotPerson=DimPerson[DimPerson['Indicator']==1]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, ".")
PlotPerson=DimPerson[DimPerson['Indicator']==2]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "o")
PlotPerson=DimPerson[DimPerson['Indicator']==3]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "+")
PlotPerson=DimPerson[DimPerson['Indicator']==4]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "^")
plt.axis('tight')
plt.title("BMI Curve")
plt.xlabel("Height(meters)")
plt.ylabel("Weight(kg)")
plt.plot()
diabetes = datasets.load_diabetes()
# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-50:]
diabetes_y_train = diabetes.target[:-30]
diabetes_y_test = diabetes.target[-50:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
```

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

Output :-

```
Row: 1077 of 1080.0
Row: 1078 of 1080.0
Row: 1079 of 1080.0
Row: 1080 of 1080.0

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

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

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

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

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

##################################
```



PRACTICAL 8

AIM :- ORGANIZING DATA

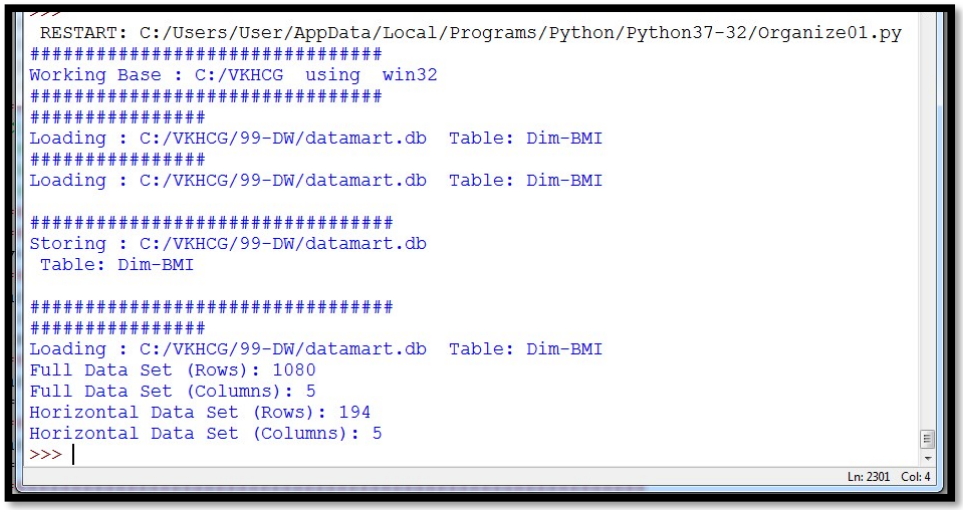
Horizontal Style

Code :-

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
 os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT PersonID,\
Height,\
Weight,\
bmi,\
Indicator\
FROM [Dim-BMI]\
WHERE \
Height > 1.5 \
and Indicator = 1\
ORDER BY \
Height,\
```

```
Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#########################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#########################################################################
sTable = 'Dim-BMI'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
#DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#########################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
```

Output :-



Vertical Style

Code :-

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
 os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
print('##################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
print('##################################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
print('##################################')
sSQL="SELECT \
Height,\
Weight,\
Indicator\
FROM [Dim-BMI];"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####################################################################
sTable = 'Dim-BMI-Vertical'
print('\n##################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n##################################')
```

```
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('##################')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
print('###################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('###################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('###################################')
```

Output :-

Island Style

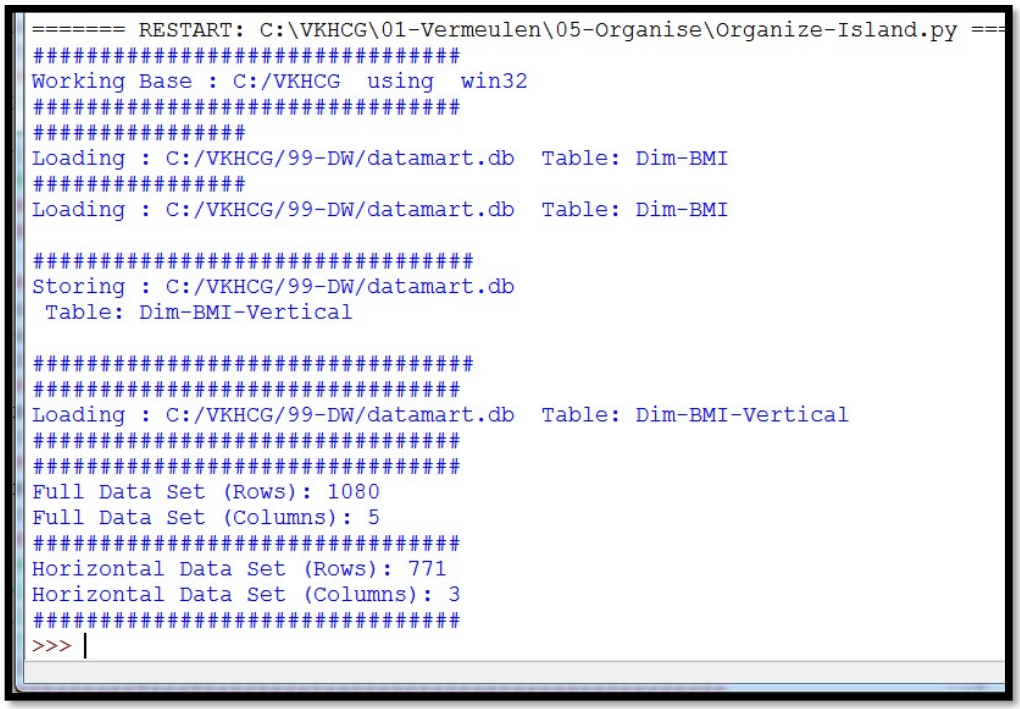
Code :-

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
```

```
if not os.path.exists(sDataWarehouseDir):
 os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
Height,\
Weight,\
Indicator\
FROM [Dim-BMI]\
WHERE Indicator > 2\
ORDER BY \
Height,\
Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####################################################################
sTable = 'Dim-BMI-Vertical'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print('#################################')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#################################')
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#####################################################################
print('#################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('#################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
```

```
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('#################################')
#####################################################################
```

Output :-



Secure Vault Style

Code :-

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
#####################################################################
Company='01-Vermeulen'
#####################################################################
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
 os.makedirs(sDataWarehouseDir)
```

```
#################################################################
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#################################################################
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#################################################################
print('#################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
Height,\
Weight,\
Indicator,\
CASE Indicator\
WHEN 1 THEN 'Pip'\
WHEN 2 THEN 'Norman'\
WHEN 3 THEN 'Grant'\
ELSE 'Sam'\
END AS Name\
FROM [Dim-BMI]\
WHERE Indicator > 2\
ORDER BY \
Height,\
Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#################################################################
sTable = 'Dim-BMI-Secure'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('#################################')
sTable = 'Dim-BMI-Secure'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#################################')
sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#################################################################
print('#################################')
```

```
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('#################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('Only Sam Data')
print(PersonFrame2.head())
print('#################################')
#########################################################################
```

Output :-

```
>>>
==== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Secure-Vault.py ====
################################
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
################################
```

Association Rule Mining

Code :-

```
import sys
import os
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
#########################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#########################################################################
Company='01-Vermeulen'
InputFileName='Online-Retail-Billboard.xlsx'
```

```
EDSAssessDir='02-Assess/01-EDS'
InputAssessDir=EDSAssessDir + '/02-Python'
########################################################################
sFileAssessDir=Base + '/' + Company + '/' + InputAssessDir
if not os.path.exists(sFileAssessDir):
os.makedirs(sFileAssessDir)
########################################################################
sFileName=Base+'/'+ Company + '/00-RawData/' + InputFileName
########################################################################
df = pd.read_excel(sFileName)
print(df.shape)
########################################################################
df['Description'] = df['Description'].str.strip()
df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]
basket = (df[df['Country'] =="France"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
########################################################################
def encode_units(x):
 if x <= 0:
  return 0
if x >= 1:
   return 1
########################################################################
basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
print(rules.head())
rules[ (rules['lift'] >= 6) &
(rules['confidence'] >= 0.8) ]
########################################################################
sProduct1='ALARM CLOCK BAKELIKE GREEN'
print(sProduct1)
print(basket[sProduct1].sum())
sProduct2='ALARM CLOCK BAKELIKE RED'
print(sProduct2)
print(basket[sProduct2].sum())
########################################################################
basket2 = (df[df['Country'] =="Germany"]
.groupby(['InvoiceNo', 'Description'])['Quantity']
.sum().unstack().reset_index().fillna(0)
.set_index('InvoiceNo'))
basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)
```

```
print(rules2[ (rules2['lift'] >= 4) &
(rules2['confidence'] >= 0.5)])
#####################################################################
print('### Done!! ##############################################')
#####################################################################
```

Output :-

```
== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Association-Rule.py ==
################################
Working Base : C:/VKHCG  using  win32
################################
(541909, 8)
                   antecedents  ... conviction
0   (ALARM CLOCK BAKELIKE PINK)  ...   3.283859
1  (ALARM CLOCK BAKELIKE GREEN)  ...   3.791383
2  (ALARM CLOCK BAKELIKE GREEN)  ...   4.916181
3    (ALARM CLOCK BAKELIKE RED)  ...   5.568878
4   (ALARM CLOCK BAKELIKE PINK)  ...   3.293135

[5 rows x 9 columns]
ALARM CLOCK BAKELIKE GREEN
340.0
ALARM CLOCK BAKELIKE RED
316.0
                       antecedents  ... conviction
0   (PLASTERS IN TIN CIRCUS PARADE)  ...   2.076984
7        (PLASTERS IN TIN SPACEBOY)  ...   2.011670
11    (RED RETROSPOT CHARLOTTE BAG)  ...   5.587746

[3 rows x 9 columns]
### Done!! ############################################
```

Create a Network Routing Diagram

Code :-

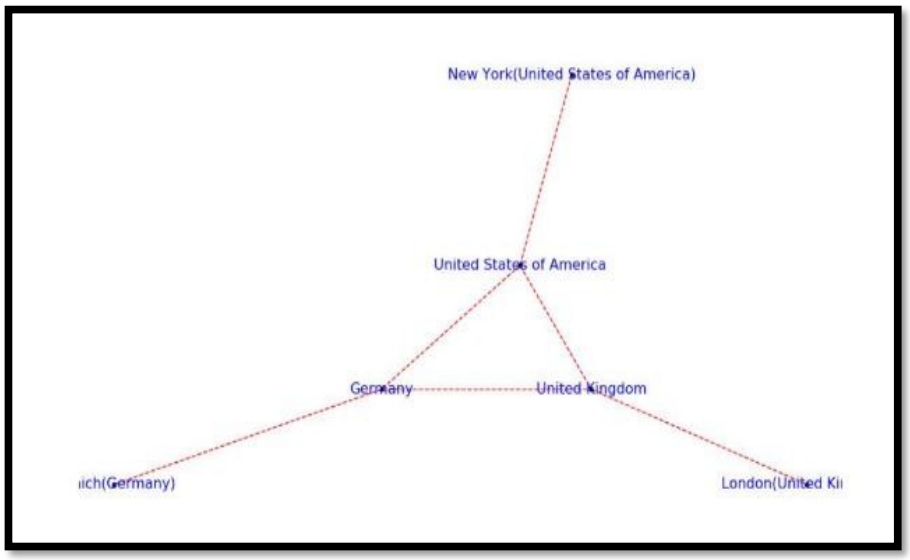
```
Create a Network Routing Diagram
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
#####################################################################
pd.options.mode.chained_assignment = None
#####################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#####################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#####################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv'
#####################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Network-Routing-
Company.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Network-Routing-
Company.png'
```

```
Company='01-Vermeulen'
##########################################################################
##########################################################################
### Import Country Data
##########################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('################################')
##########################################################################
print(CompanyData.head())
print(CompanyData.shape)
##########################################################################
G=nx.Graph()
for i in range(CompanyData.shape[0]):
 for j in range(CompanyData.shape[0]):
Node0=CompanyData['Company_Country_Name'][i]
Node1=CompanyData['Company_Country_Name'][j]
if Node0 != Node1:
 G.add_edge(Node0,Node1)
for i in range(CompanyData.shape[0]):
Node0=CompanyData['Company_Country_Name'][i]
Node1=CompanyData['Company_Place_Name'][i] + '('+
CompanyData['Company_Country_Name'][i] + ')'
if Node0 != Node1:
    G.add_edge(Node0,Node1)
print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
##########################################################################
sFileName=Base + '/' + Company + '/' + sOutputFileName1
print('################################')
print('Storing :',sFileName)
print('################################')
nx.write_gml(G, sFileName)
##########################################################################
sFileName=Base + '/' + Company + '/' + sOutputFileName2
print('################################')
print('Storing Graph Image:',sFileName)
print('################################')
plt.figure(figsize=(15, 15))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
print('################################')
```

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

Output :-

```
== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Association-Rule.py ==
################################
Working Base : C:/VKHCG  using  win32
################################
(541909, 8)
                    antecedents  ... conviction
0   (ALARM CLOCK BAKELIKE PINK)  ...   3.283859
1  (ALARM CLOCK BAKELIKE GREEN)  ...   3.791383
2  (ALARM CLOCK BAKELIKE GREEN)  ...   4.916181
3    (ALARM CLOCK BAKELIKE RED)  ...   5.568878
4   (ALARM CLOCK BAKELIKE PINK)  ...   3.293135

[5 rows x 9 columns]
ALARM CLOCK BAKELIKE GREEN
340.0
ALARM CLOCK BAKELIKE RED
316.0
                       antecedents  ... conviction
0   (PLASTERS IN TIN CIRCUS PARADE)  ...   2.076984
7        (PLASTERS IN TIN SPACEBOY)  ...   2.011670
11    (RED RETROSPOT CHARLOTTE BAG)  ...   5.587746

[3 rows x 9 columns]
### Done!! ##########################################
```



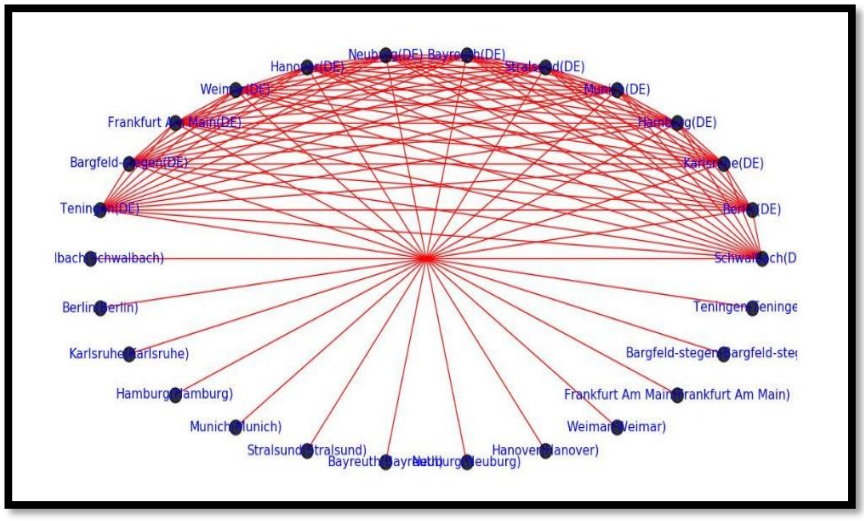
Picking Content for Billboards

Code :-

```
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
#################################################################
pd.options.mode.chained_assignment = None
#################################################################
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
#################################################################
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
#################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-DE-Billboard-Visitor.csv'
#################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Billboards.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Billboards.png'
Company='02-Krennwallner'
#################################################################
#################################################################
### Import Company Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
BillboardDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
print('#################################')
#################################################################
print(BillboardDataRaw.head())
print(BillboardDataRaw.shape)
BillboardData=BillboardDataRaw
sSample=list(np.random.choice(BillboardData.shape[0],20))
#################################################################
G=nx.Graph()
for i in sSample:
 for j in sSample:
  Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['BillboardCountry'][i]
+ ')'
Node1=BillboardData['BillboardPlaceName'][j] + '('+ BillboardData['BillboardCountry'][i] +
')'
if Node0 != Node1:
G.add_edge(Node0,Node1)
for i in sSample:
```

```
 Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorPlaceName'][i]
+ ')'
Node1=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorCountry'][i] + ')'
if Node0 != Node1:
 G.add_edge(Node0,Node1)
print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
#####################################################################
sFileName=Base + '/02-Krennwallner/' + sOutputFileName1
print('###################################')
print('Storing :',sFileName)
print('###################################')
nx.write_gml(G, sFileName)
#####################################################################
sFileName=Base + '/02-Krennwallner/' + sOutputFileName2
print('###################################')
print('Storing Graph Image:',sFileName)
print('###################################')
plt.figure(figsize=(15, 15))
pos=nx.circular_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=150, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='solid')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
print('###################################')
print('### Done!! ######################')
print('###################################')
#####################################################################
```

Output :-



Create a Delivery Route

Code :-

```
import sys
import os
import pandas as pd
Base='C:/Users/Abdul Razzaque/Desktop/IMP/MSC-IT/practical-data-science/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sInputFileName='02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.txt'
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Routes.csv'
Company='03-Hillman'
### Import Routes Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
RouteDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, sep='|',
encoding="latin-1")
print('##################################')
RouteStart=RouteDataRaw[RouteDataRaw['StartAt']=='WH-KA13']
#########################################################################
RouteDistance=RouteStart[RouteStart['Cost']=='DistanceMiles']
RouteDistance=RouteDistance.sort_values(by=['Measure'], ascending=False)
#########################################################################
RouteMax=RouteStart["Measure"].max()
RouteMaxCost=round((((RouteMax/1000)*1.5*2)),2)
print('##################################')
print('Maximum (£) per day:')
print(RouteMaxCost)
print('##################################')
RouteMean=RouteStart["Measure"].mean()
RouteMeanMonth=round((((RouteMean/1000)*2*30)),6)
print('##################################')
print('Mean per Month (Miles):')
print(RouteMeanMonth)
print('##################################')
```

Output :-

```
======== RESTART: C:\VKHCG\03-Hillman\05-Organise\Organise-Routes.py ========
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/03-Hillman/02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.
txt
################################
################################
################################
Maximum (£) per day:
21.82
################################
################################
Mean per Month (Miles):
21.56191
################################
```

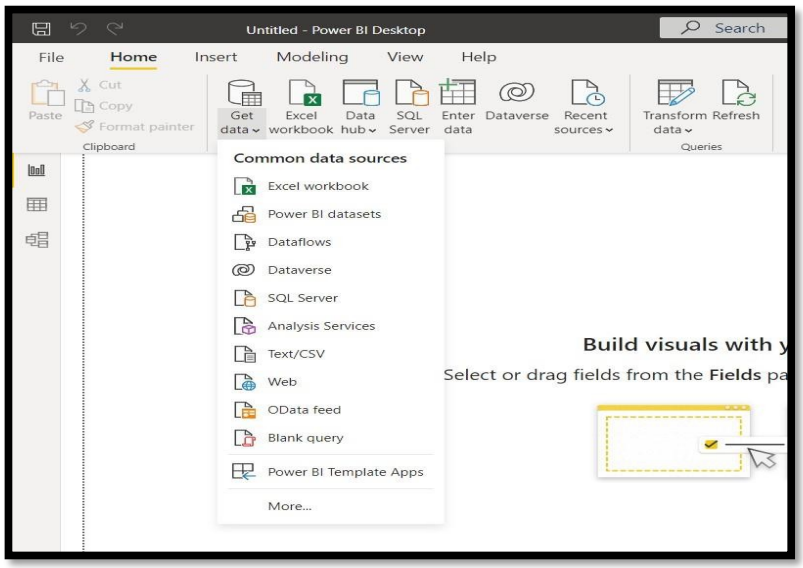
PRACTICAL 9

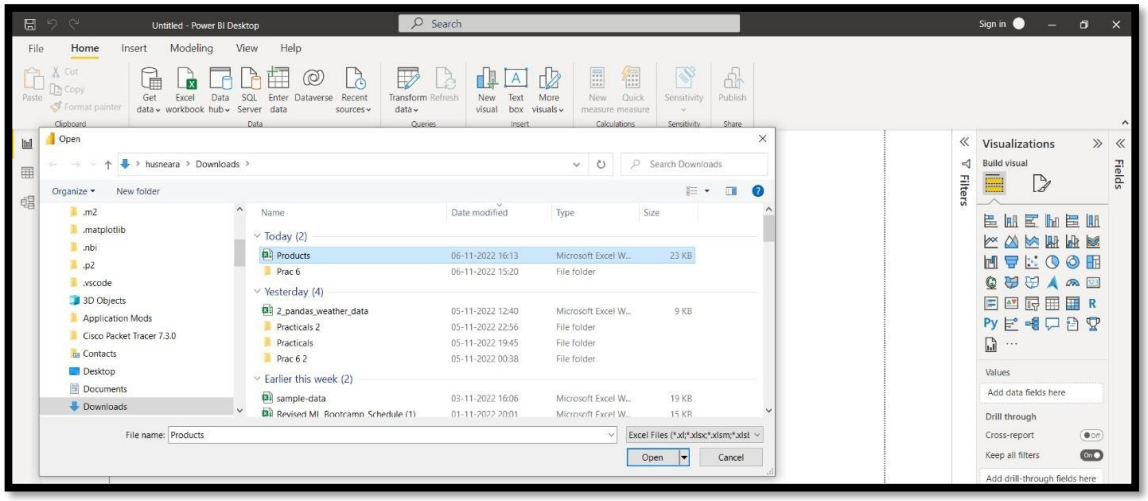
AIM :- DATA VISUALIZATION WITH POWER BI

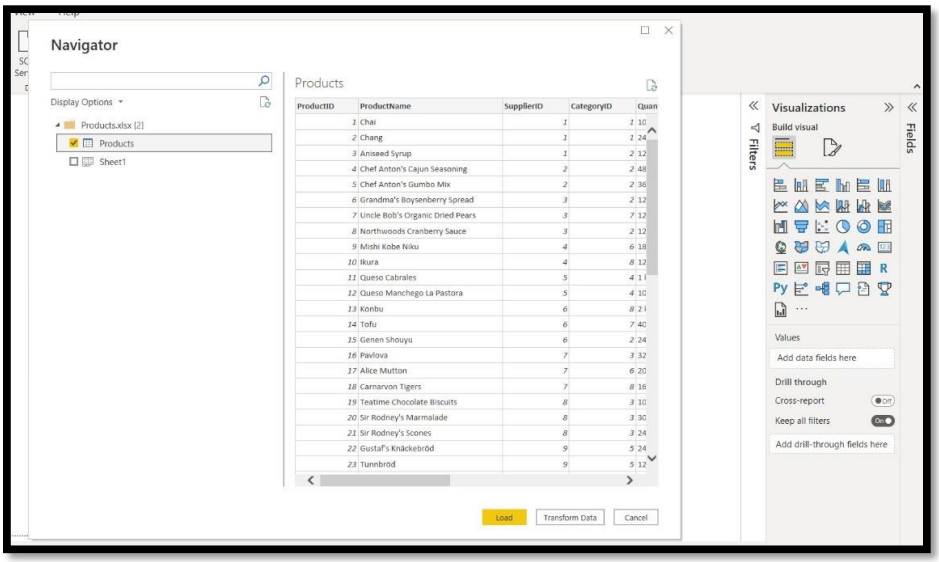
Task 1: Case Study - Sales Data

Steps:

1. Launch Power BI Desktop
2. From the home ribbon, select Get Data.
3. File>Excel and select connect
4. In the dialog box, the Product.xlsx file.

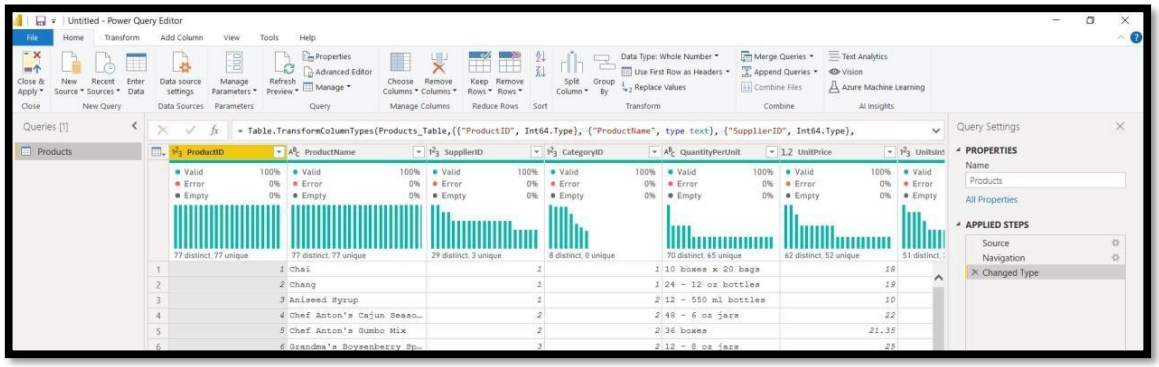


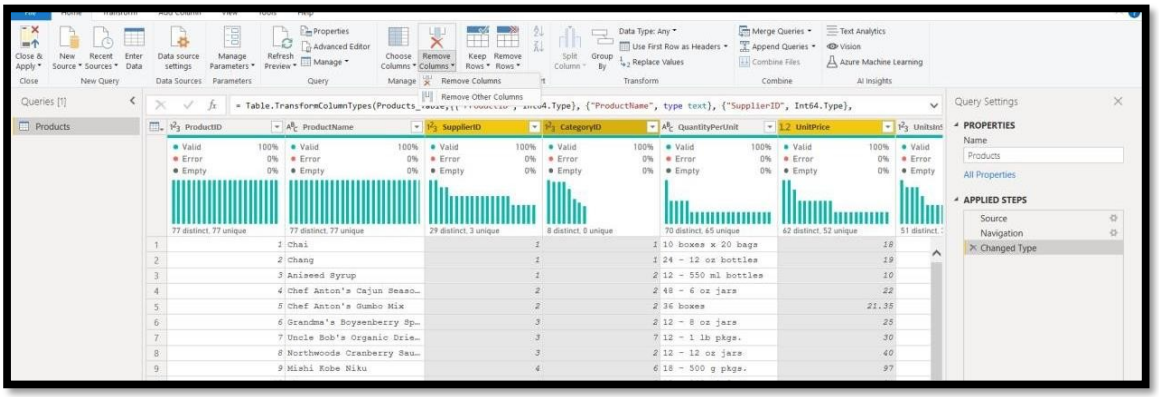


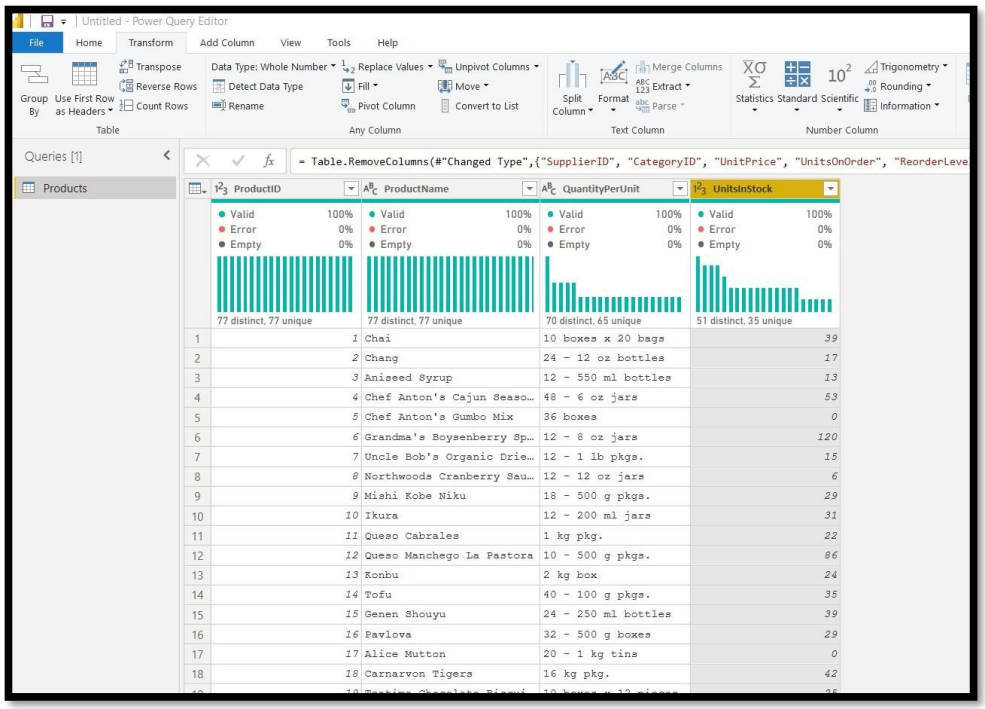


You can also open the Query Editor by selecting Edit Queries from the home ribbon in PowerBI Desktop.

5. In Query Editor, select others Column Except the **ProductID, ProductName,QuantityPerUnit,** and **UnitsInStock** columns.
6. Select Remove Columns > Right-click on a Remove Column.

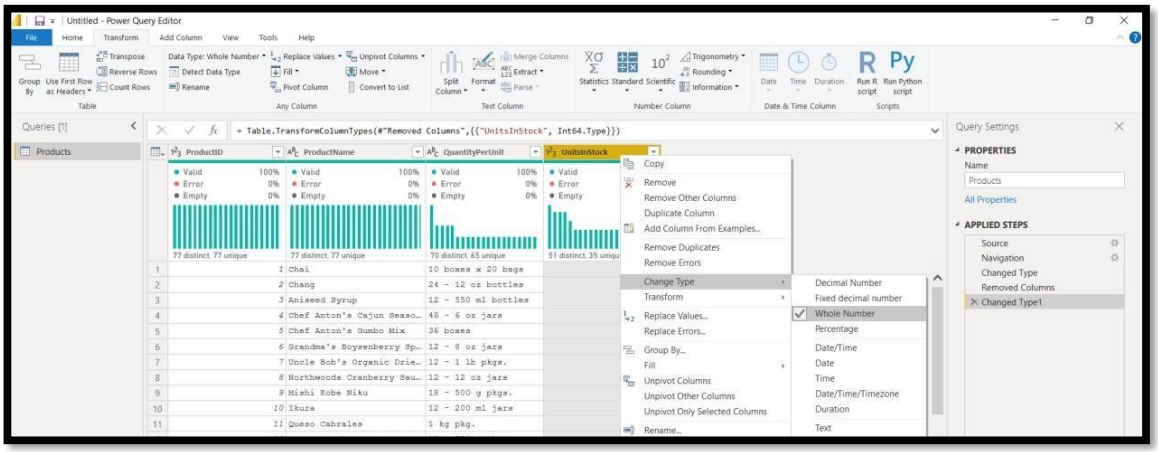






Change the data type of the UnitsInStock column.

7. Select the **UnitsInStock** column.
8. Select the **Data Type** drop-down button in the **home** ribbon.
9. If not already a **Whole Number**, select Whole Number for data type from the dropdown.

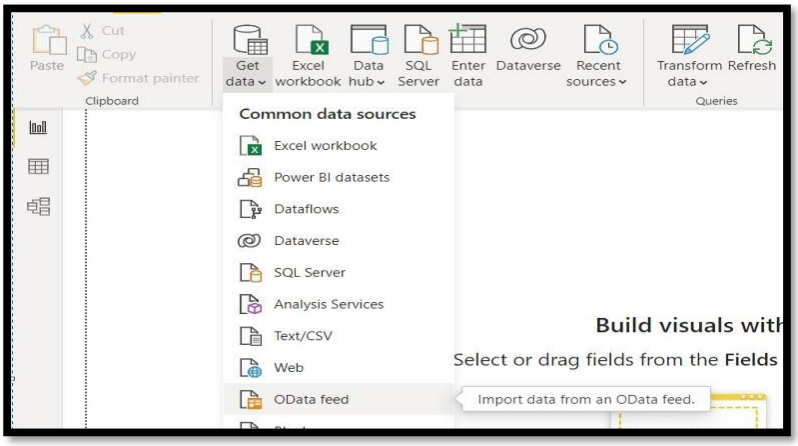


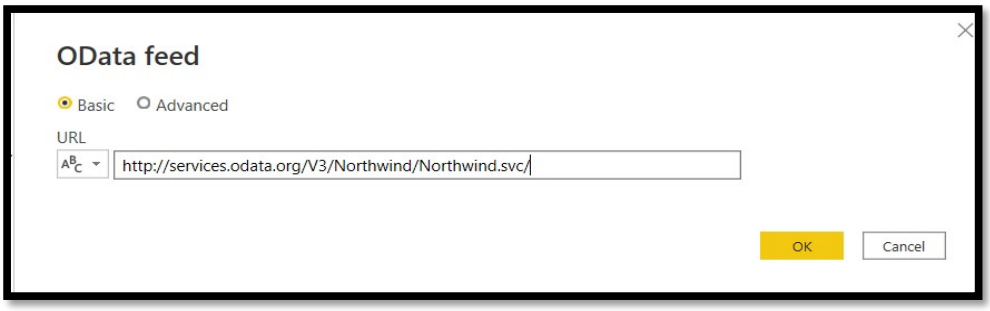
Task 2: Import Order Data from and OData Feed

Connect to an OData feed

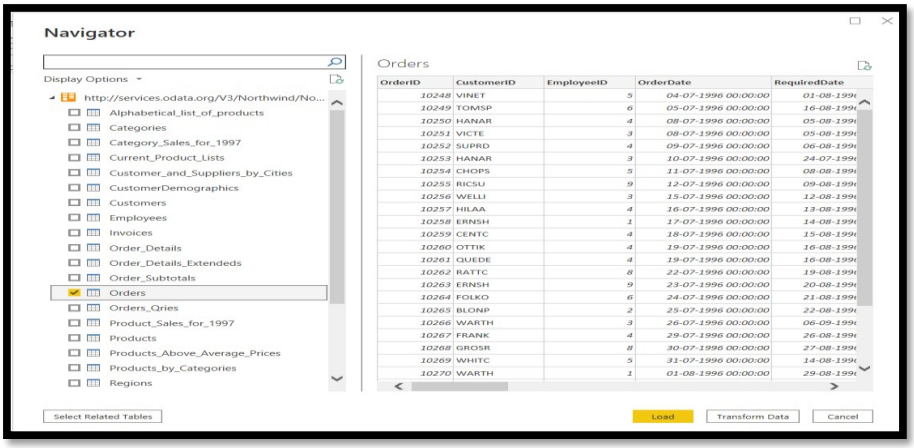
Steps:

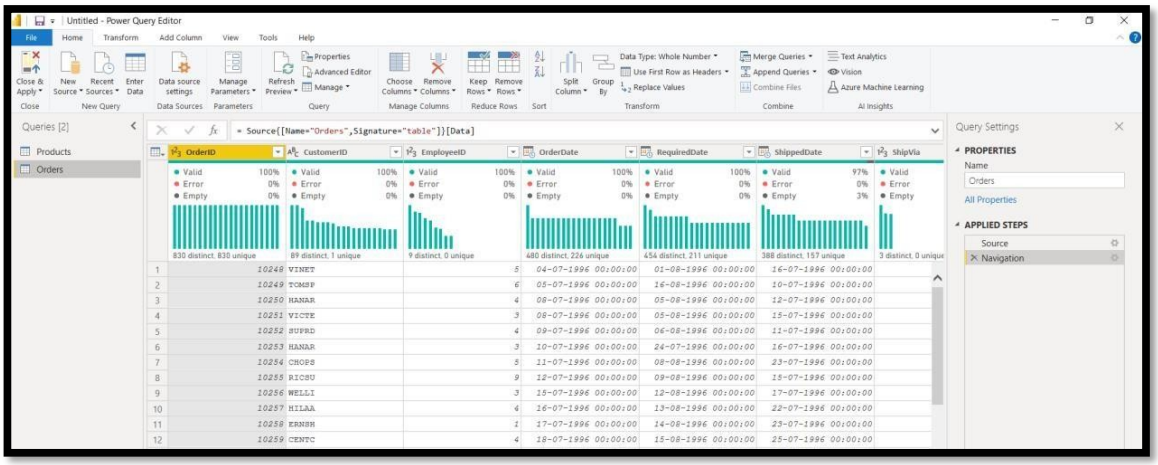
1. From the Home ribbon tab in Query Editor, select Get Data.
2. Browse to the OData Feed data source.
3. In the OData Feed dialog box, paste the URL for the Northwind OData feed. Select OK.





Step 4. Select Order Table → Click on Load.

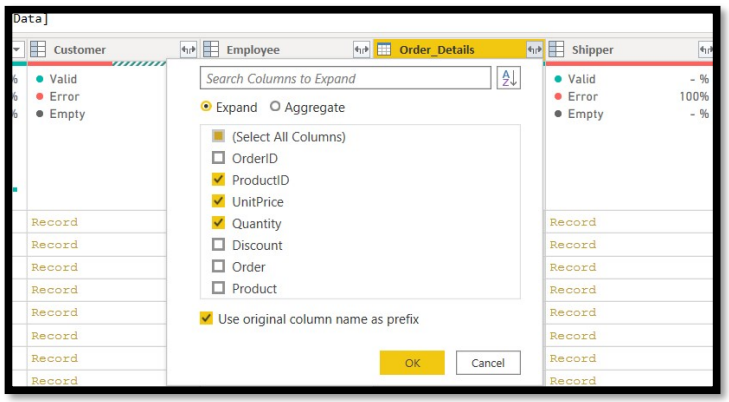




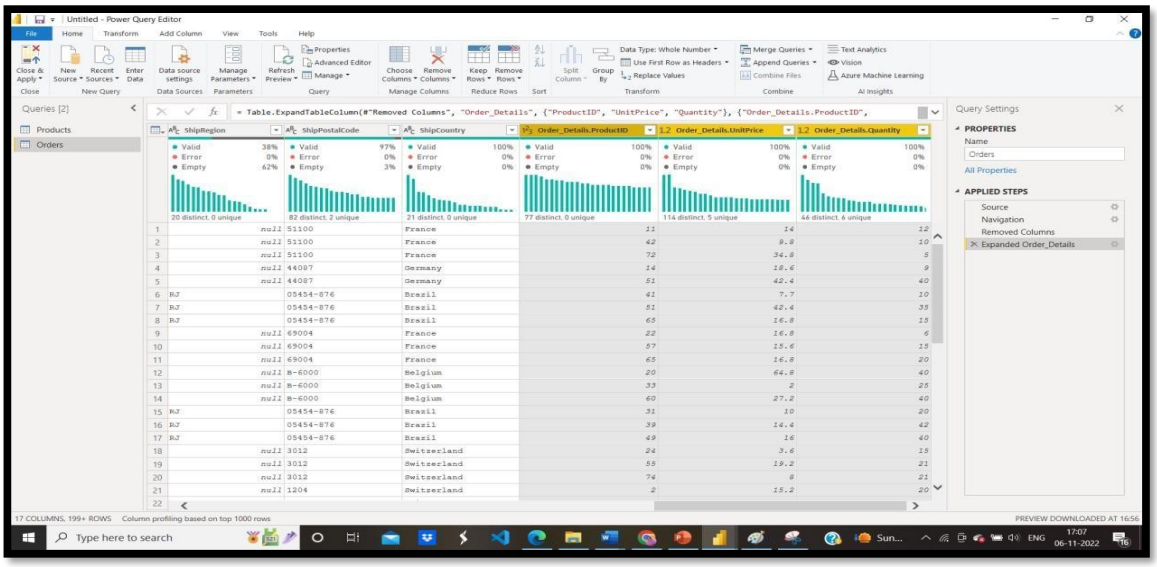
Expand the Order_Details tableSteps:

Expand the **Order_Details** table that is related to the **Orders** table, to combine the **ProductID, UnitPrice,** and **Quantity** columns from **Order_Details** into the **Orders** table.

1. In the **Query View,** scroll to the **Order_Details** column.
2. In the **Order_Details** column, select the expand icon ().
3. Select (Select All Columns) to clear all columns.
4. Select **ProductID, UnitPrice,** and **Quantity.**



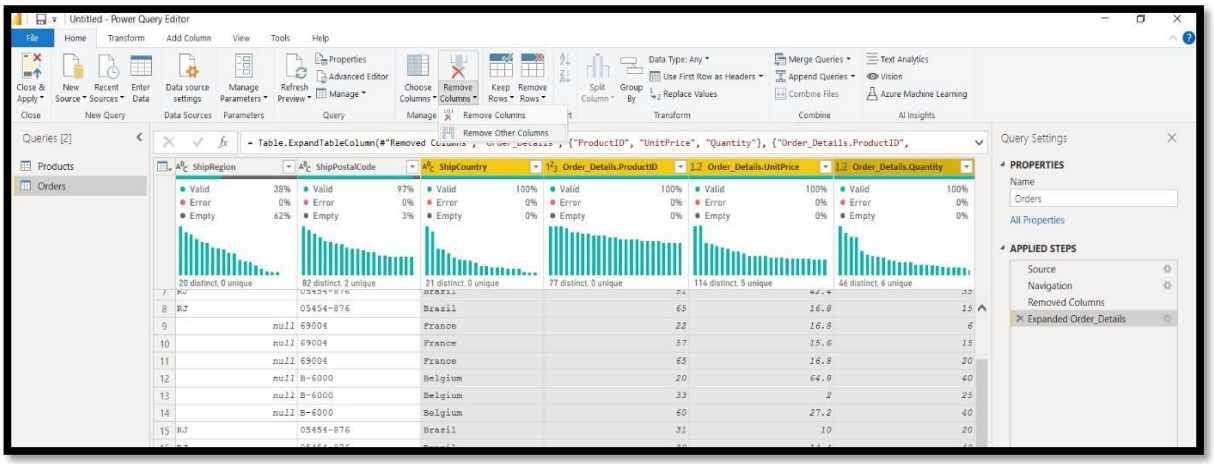
5. Click OK.

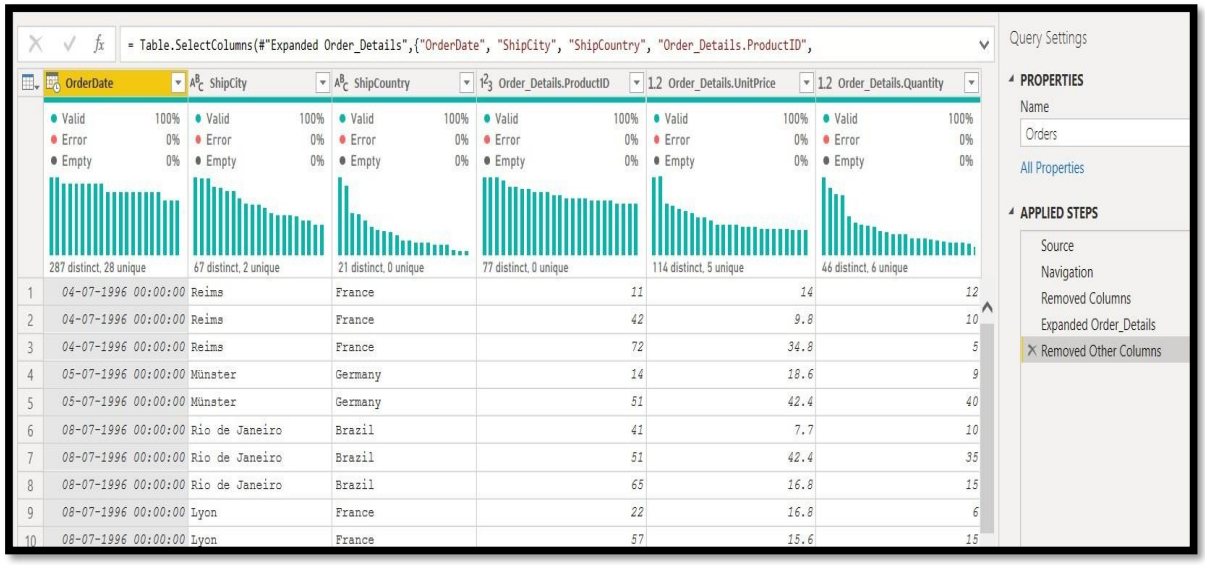


Remove other columns to only display columns of

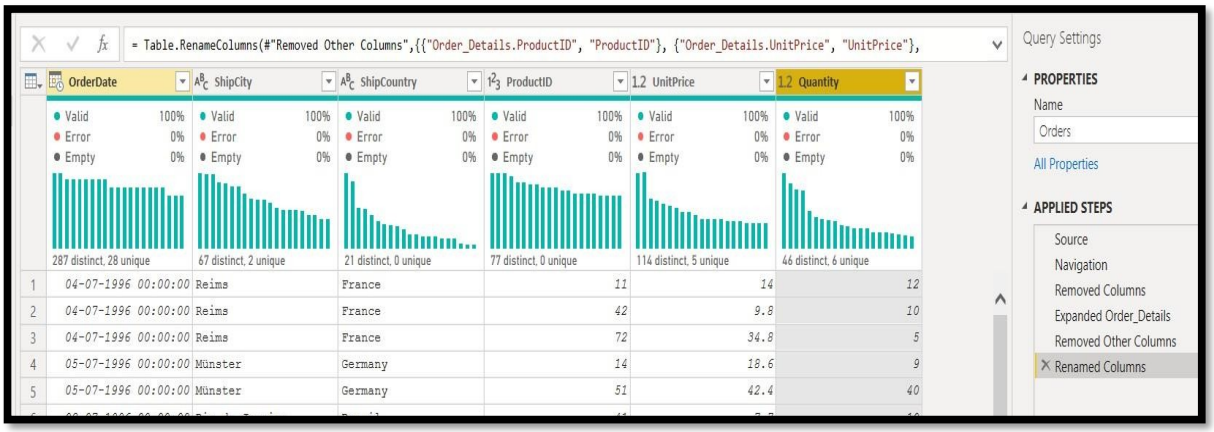
interest Steps:

1. In this step you remove all columns except **OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice,** and **Order_Details.Quantity** columns.
2. Now that only the columns we want to remove are selected, right-click on anyselected column header and click Remove Columns.



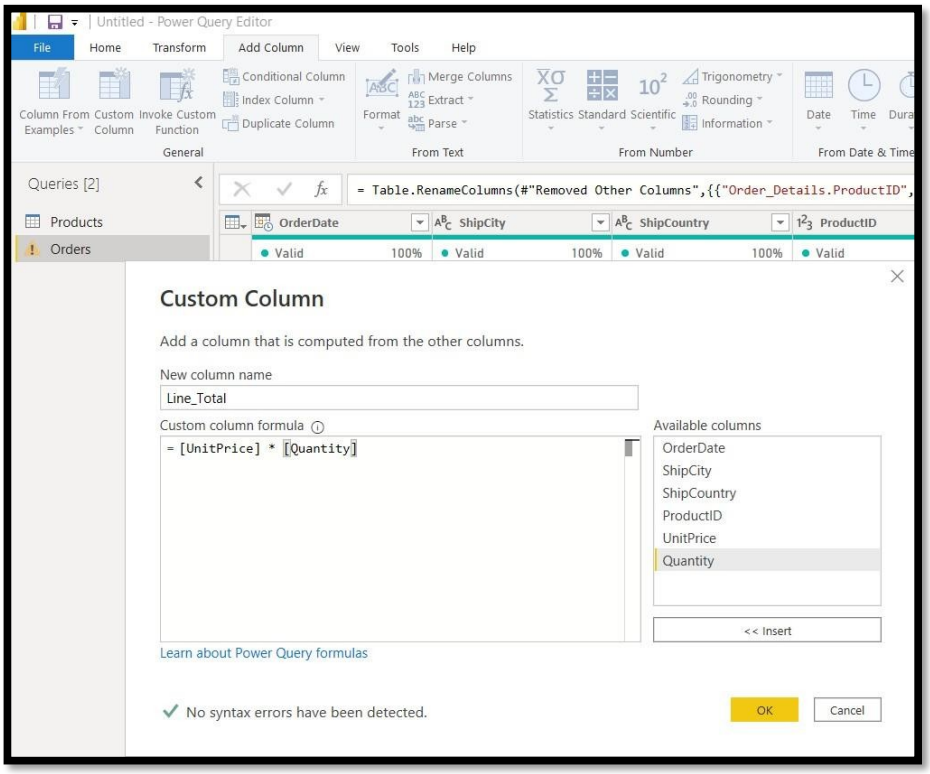


3. Rename the column **Order_Details.ProductID, Order_Details.UnitPrice,** and Order_Details.Quantity into ProductID , UnitPrice, Quantity.



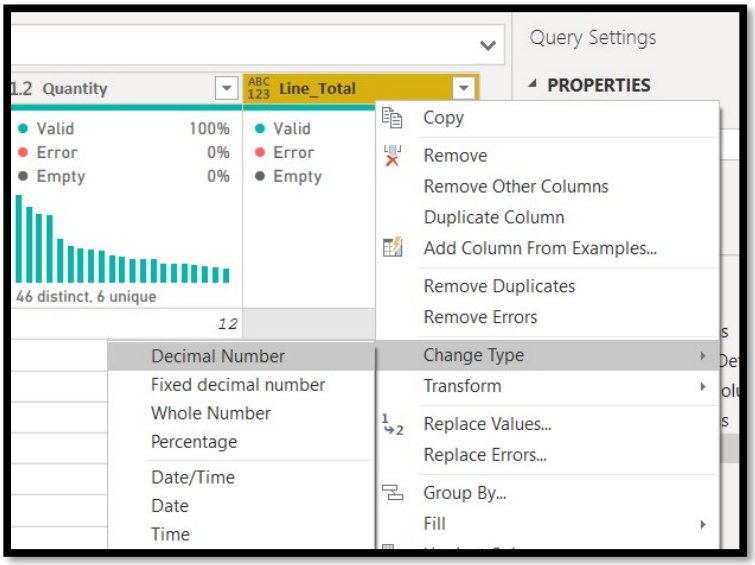
Calculate the line total for each Order_Details row

1. In the **Add Column** ribbon tab, click **Add Custom Column**.
2. In the Add Custom Column dialog box, in the Custom Column Formulatextbox, enter **[UnitPrice] * [Quantity].**
3. In the New column name textbox, enter Line_Total.



Set the Data Type of the Line_Total fieldSteps:

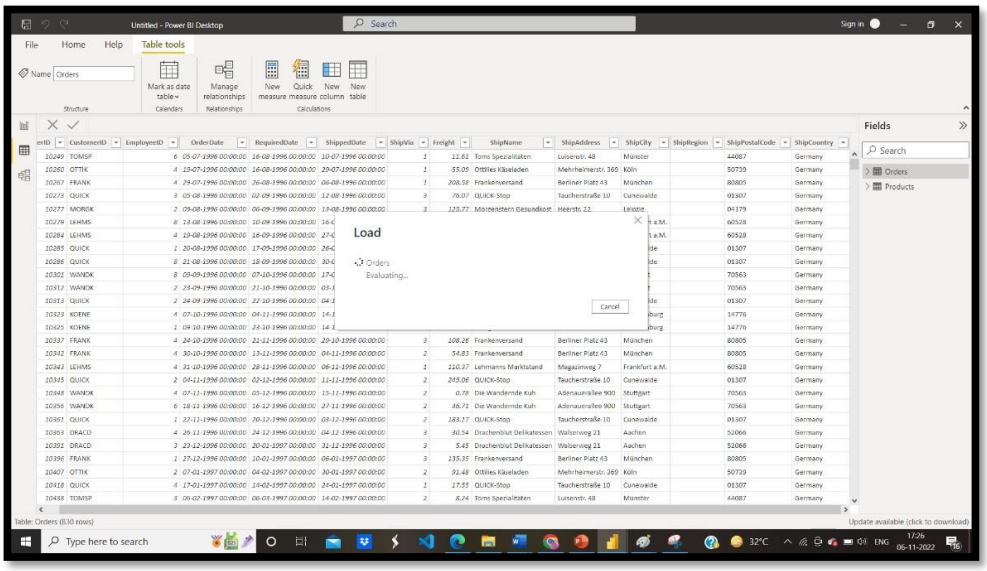
1. Right click the **Line_Total** column.
2. Select change type and choose **Decimal Number**.
3. Reorder the Line_Total Column .drag the **Line_Total** column to the left, after ShipCountry.

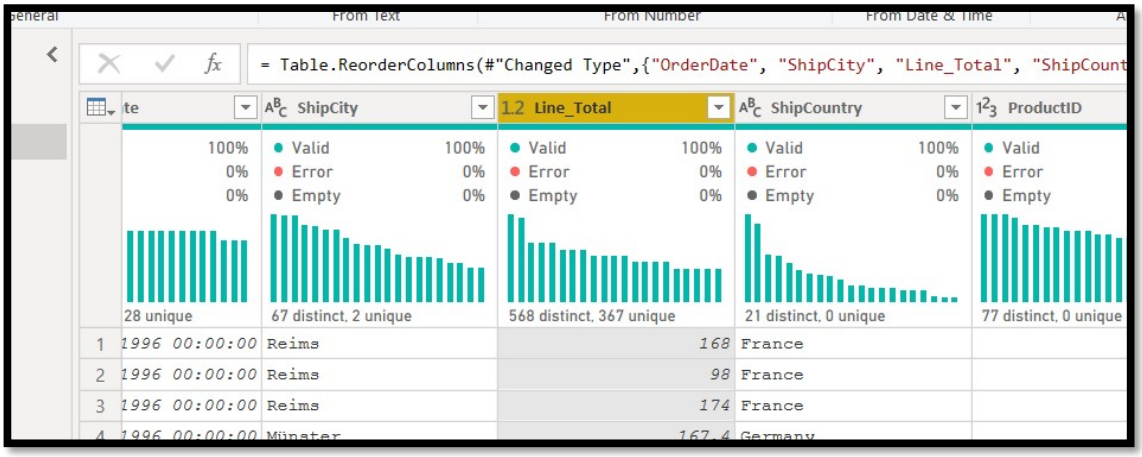


Task 3: Combine the Products andTotal Sales Queries. Confirm the relationship between Products and Total SalesSteps:

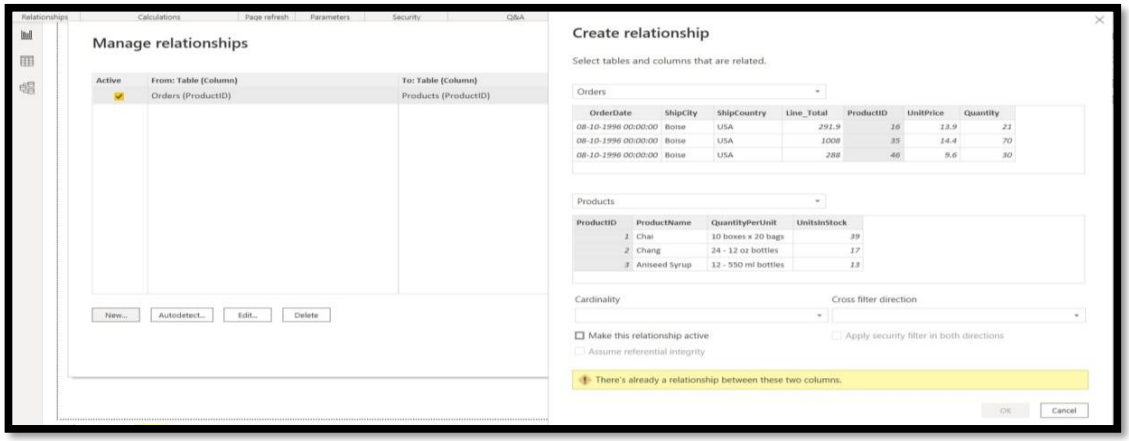
1. From the **home** ribbon of Query Editor, select **Close & Load.**
2. Power BI Desktop loads the data from the two queries.

3. Once the data is loaded, select the **Manage Relationships** button **home** ribbon.

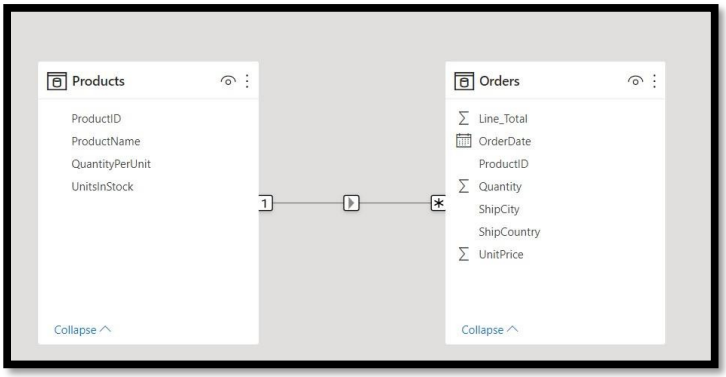




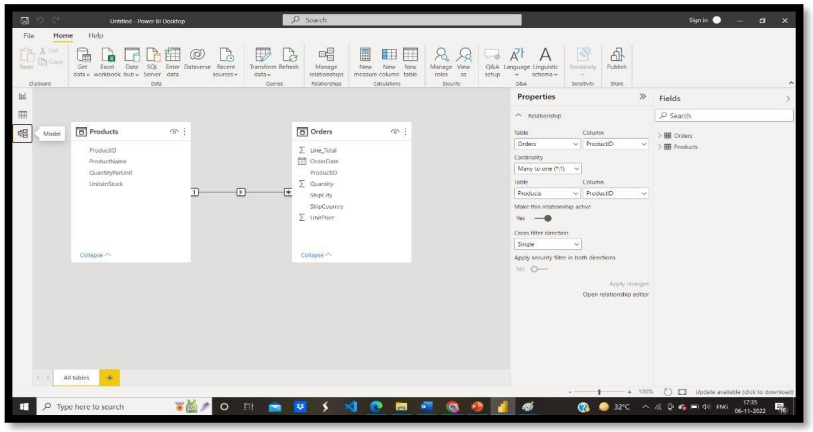
4. When we attempt to create the relationship, we see that one already exists! As Shown in the **Create Relationship** dialog (by the shaded columns), the **ProductsID** fields ineach query already have an established relationship.

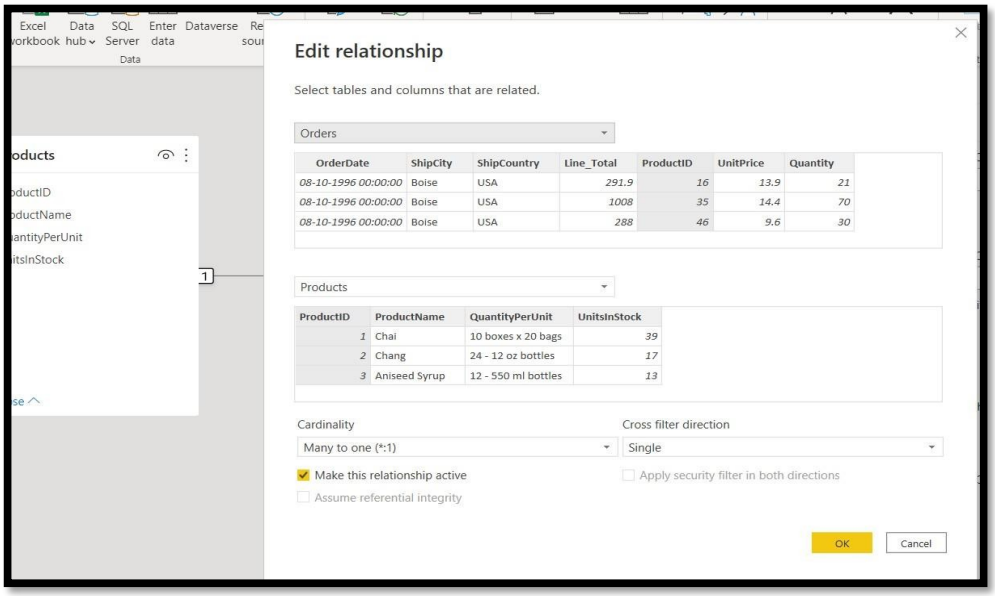


5. Which visualizes the relationship between the queries.



6. When you double-click the arrow on the line that connects the to queries, an EditRelationship dialog or In Right side there is Properties in which also you can Edit the relationship.

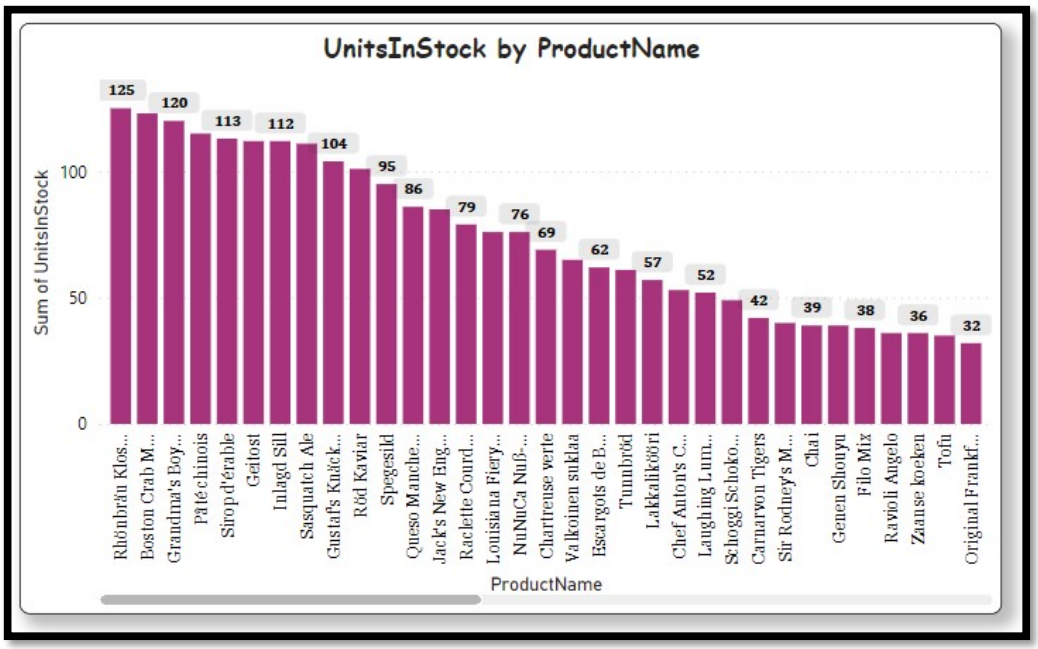




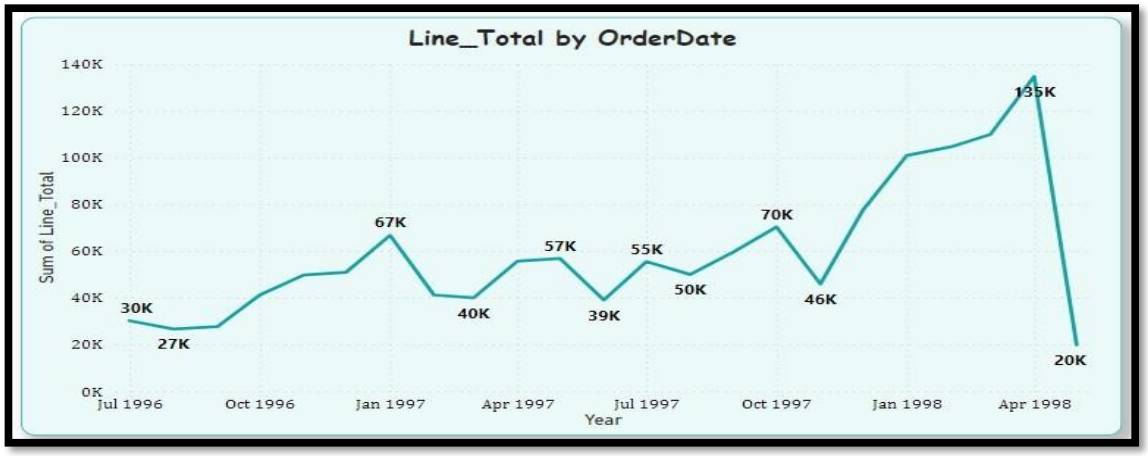
Task 4: Build visuals using your Data

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

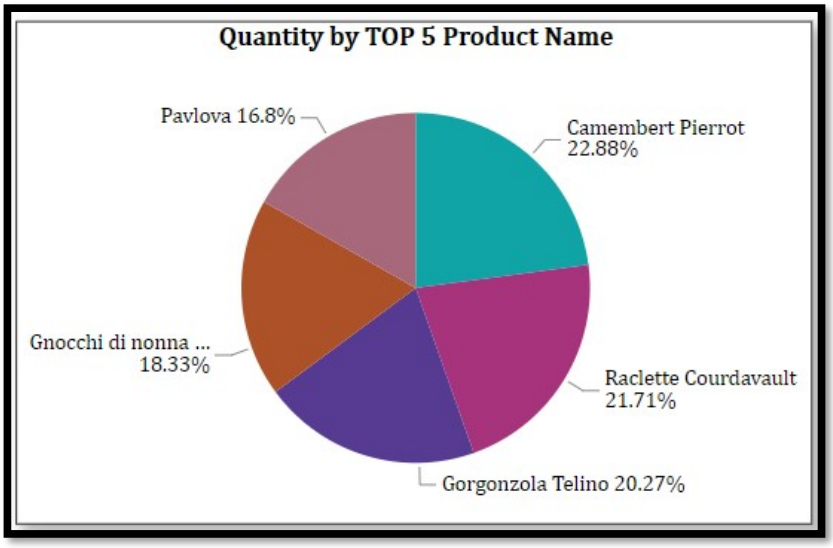
1. Drag UnitsInStock from the Field pane (the Fields pane is along the right of the screen) onto a blank space on the canvas. A Table visualization is created. Next, dragProductName to the Axis box, found in the bottom half of the Visualizations pane. Then we then select Sort By > UnitsInStock using the skittles in the top right corer ofthe visualization.



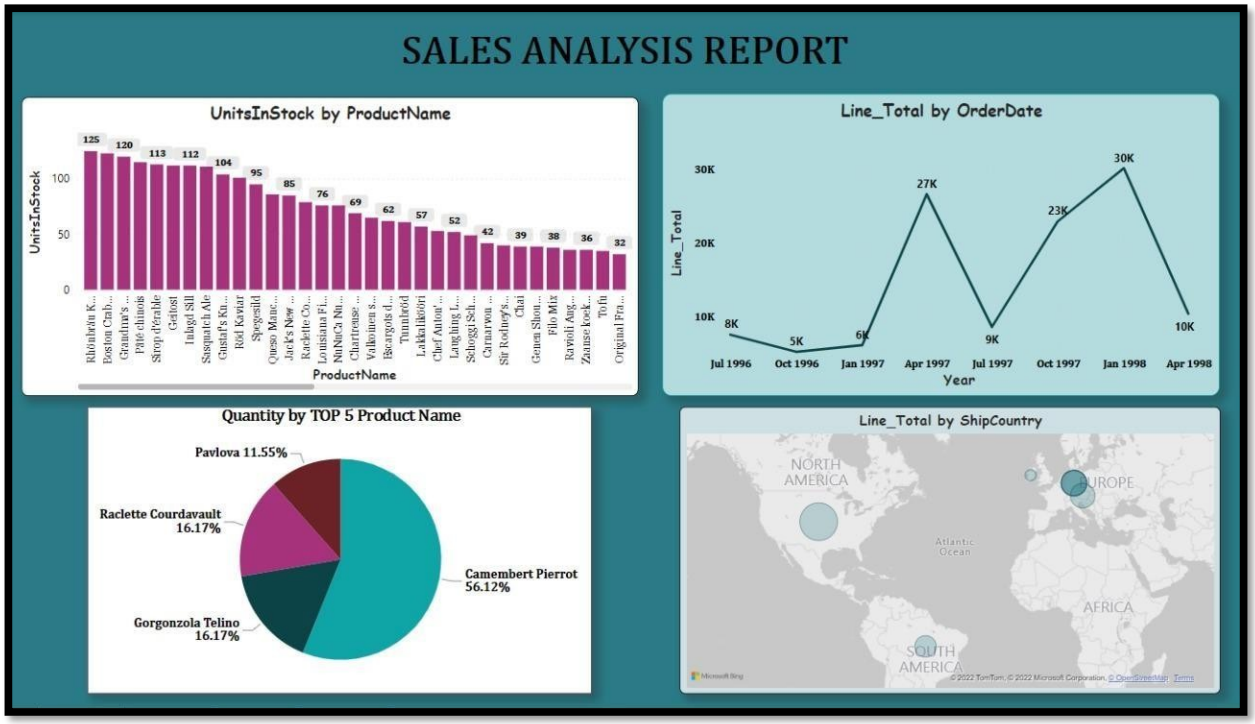
2. Drag OrderDate to the canvas beneath the first chart, then drag Line_Total (again, from the Fields pane) onto the visual, then select Line Chart. The following visualization is created.



3. Next, drag ProductName to a space on the canvas in the top right. Now drag Quantityto the Values field. Select the Pie Chart. I want to display Quantity only for Top 5 Products. So Go to Fields, Under Fields you will see ProductName. Click on the ProductName → Select Filter type→Show item select Top→By value drag Quantity→ click on Apply Filter.



Step 2: Interact with your report visuals to analyze further



FINAL REPORT AFTER APPLYING ALL FILTERS AND FROMATTING THE VISUALS