# **Practical 1**

**AIM: Write Python program to convert files from different formats to HORUS format .**

## **Text Delimited CSV format to HORUS format.**

**CODE:**

#Utility Start CSV to Horus

#standard tools

#==========================================================

import pandas as pd

# Input Agreement ============================================

print('NAME: Ayush Patel ')

print('SAP ID: 53004230035')

sInputFileName='C:/Data Science/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('CountryName', 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:/Data Science/HorusToCsv.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('CSV to HORUS - Done')

# Utility done ===============================================

## **XML format to HORUS format.**

**CODE:**

# Utility Start CSV to HORUS =================================

# Standard Tools

#=============================================================

import pandas as pd

import xml.etree.ElementTree as ET

#=============================================================

def df2xml(data):

header = data.columns

root = ET.Element('root')

for row in range(data.shape[0]):

entry = ET.SubElement(root,'entry')

for index in range(data.shape[1]):

schild=str(header[index])

child = ET.SubElement(entry, schild)

if str(data[schild][row]) != 'nan':

child.text = str(data[schild][row])

else:

child.text = 'n/a'

entry.append(child)

result = ET.tostring(root)

return result

#=============================================================

def xml2df(xml\_data):

root = ET.XML(xml\_data)

all\_records = []

for i, child in enumerate(root):

record = {}

for subchild in child:

record[subchild.tag] = subchild.text

all\_records.append(record)

return pd.DataFrame(all\_records)

#=============================================================

# Input Agreement ============================================

#=============================================================

sInputFileName='C:/Data Science/Country\_Code.xml'

InputData = open(sInputFileName).read()

print('=====================================================')

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

print('=====================================================')

print(InputData)

print('=====================================================')

#=============================================================

# Processing Rules ===========================================

#=============================================================

ProcessDataXML=InputData

# XML to Data Frame

ProcessData=xml2df(ProcessDataXML)

# Remove columns ISO-2-Code and ISO-3-CODE

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)

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

# Rename Country and ISO-M49

ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)

ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

# Set new Index

ProcessData.set\_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True)

print('=====================================================')

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

print('=====================================================')

print(ProcessData)

print('=====================================================')

#=============================================================

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

#=============================================================

OutputData=ProcessData

sOutputFileName='C:/Data Science/xmltocsv.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('=====================================================')

print('XML to HORUS - Done')

print('=====================================================')

print('NAME: Ayush Patel ')

print('SAP ID: 53004230035')

## **JSON to HORUS format.**

**CODE:**

import pandas as pd

#Input Agreement

sInputFileName='C:/Data Science/Country\_Code.json'

InputData=pd.read\_json(sInputFileName,orient='index',encoding="latin-1")

print('Input Data Values')

print(InputData)

#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 currency number

ProcessData.sort\_values('CountryName',axis=0,ascending=False,inplace=True)

print('Process Data Values')

print(ProcessData)

#Output Agreement

OutputData=ProcessData

sOutputFileName='C:\Data Science/HORUS-JSON-Country.csv'

OutputData.to\_csv(sOutputFileName,index=False)

print('JSON to HORUS-Done')

print('======= Ayush Patel : 53004230035')

## **MySQL Database to HORUS format.**

**CODE:**

# Utility Start Database to HORUS

import pandas as pd

import sqlite3 as sq

# Input Agreement

sInputFileName='C:/Data Science/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)

# 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)

# Output Agreement

OutputData=ProcessData

sOutputFileName='C:/Data Science/DATABSE-HORUS-Country.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('Database to HORUS - Done')

print("Ayush Patel - 53004230035")

# Utility done

## **Picture (JPEG) to HORUS format.**

**CODE:**

from matplotlib.pyplot import imread

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

# Input Agreement

sInputFileName='C:/Data Science/Angus.jpg'

InputData = imread(sInputFileName)

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

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

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

# Processing Rules

ProcessRawData=InputData.flatten()

y=InputData.shape[2] + 3

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('Process Data Values ')

plt.imshow(InputData)

plt.show()

# Output Agreement

OutputData=ProcessData

print('Storing File')

sOutputFileName='C:/Data Science/HORUS-Picture.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('Picture to HORUS - Done')

print("Ayush Patel - 53004230035")

## **Audio to HORUS format.**

**CODE:**

from scipy.io import wavfile

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

print('Ayush Patel - 53004230035')

def show\_info(aname,a,r):

print('Audio: ',aname)

print()

print('Rate: ',r)

print()

print('shape: ',a.shape)

print('dtype: ',a.dtype)

print('min,max: ',a.min(),a.max())

print()

print(plot\_info(aname,a,r))

def plot\_info(aname,a,r):

Title= 'Signal Wave- '+ aname +' at '+ str(r) +' hz '

plt.title(Title)

Legend= []

for c in range(a.shape[1]):

Label = 'Ch' + str(c+1)

Legend = Legend + [str(c+1)]

plt.plot(a[:,c],label=Label)

plt.legend(Legend)

plt.show()

InputFileName = 'C:/Data Science/2ch-sound.wav'

InputRate,InputData = wavfile.read(InputFileName)

show\_info('2 channel',InputData,InputRate)

ProcessData = pd.DataFrame(InputData)

Columns = ['Ch1','Ch2']

ProcessData.columns=Columns

OutputData = ProcessData

OutputFileName = 'C:/Data Science/Horus-Audio-2ch.csv'

OutputData.to\_csv(OutputFileName,index=False)

InputFileName = 'C:/Data Science/4ch-sound.wav'

InputRate,InputData = wavfile.read(InputFileName)

show\_info('4 Channel',InputData,InputRate)

ProcessData = pd.DataFrame(InputData)

Columns = ['Ch1','Ch2','Ch3','Ch4']

ProcessData.columns = Columns

OutputData = ProcessData

OutputFileName = 'C:/Data Science/Horus-Audio-4ch.csv'

OutputData.to\_csv(OutputFileName,index=False)

InputFileName = 'C:/Data Science/6ch-sound.wav'

InputRate,InputData = wavfile.read(InputFileName)

show\_info('6 Channel',InputData,InputRate)

ProcessData = pd.DataFrame(InputData)

Columns = ['Ch1','Ch2','Ch3','Ch4','Ch5','Ch6']

ProcessData.columns = Columns

OutputData = ProcessData

OutputFileName = 'C:/Data Science/Horus-Audio-6ch.csv'

OutputData.to\_csv(OutputFileName,index=False)

InputFileName = 'C:/Data Science/8ch-sound.wav'

InputRate,InputData = wavfile.read(InputFileName)

show\_info('8 Channel',InputData,InputRate)

ProcessData = pd.DataFrame(InputData)

Columns = ['Ch1','Ch2','Ch3','Ch4','Ch5','Ch6','Ch7','Ch8']

ProcessData.columns = Columns

OutputData = ProcessData

OutputFileName = 'C:/Data Science/Horus-Audio-8ch.csv'

OutputData.to\_csv(OutputFileName,index=False)

# **Practical 2**

**AIM: Utilities and Auditing**

## **Fixer Utility.**

**CODE:**

import string

import datetime as dt

#1 Removing leading or lagging spaces from a data entry

print('1.Removing leading or lagging spaces from a data entry');

baddata="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(2022,11,10)

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)

print('Ayush Patel - 53004230035')

## **Data Binning or Bucketing.**

**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('Probablity 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:/Data Science/DU-Histogram.png'

fig.savefig(sPathFig)

plt.show()

print('Ayush Patel - 53004230035')

## **Averaging of Data.**

**CODE:**

import pandas as pd

InputFileName='IP\_DATA\_CORE.csv'

OutputFileName='Retrieve\_Router\_Location.csv'

Base='C:/Data Science/'

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

print('==================================')

sFileName = Base + 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)

print('Ayush PAtel - 53004230035')

## **Outlier Detection.**

**CODE:**

import pandas as pd

InputFileName='IP\_DATA\_CORE.csv'

OutputFileName='Retrieve\_Router\_Location.csv'

Base='C:/Data Science/'

print('Working Base:',Base)

print('==================================')

sFileName = Base + 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('AllData')

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)

print('Ayush Patel - 53004230035')

## **Logging.**

**CODE:**

import pandas as pd

import sys

import os

import logging

import uuid

import shutil

import time

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

if sys.platform == 'linux':

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

else:

Base=’C:/’

print(‘Ayush Patel – 53004230035’)

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

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.')

# **Practical 3**

**AIM: Retrieve Superstep**

> library(readr)

> IP\_DATA\_ALL<- read.csv('C:/Users/DELL/OneDrive/Desktop/MSc IT-1/DataSciencePractical/VKHCG/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv')

> View(IP\_DATA\_ALL)

> library(tibble)

> IP\_DATA\_ALL<-as\_tibble(IP\_DATA\_ALL)

> spec(IP\_DATA\_ALL)

> set\_tidy\_names(IP\_DATA\_ALL,syntactic=TRUE,quiet=FALSE)

> IP\_DATA\_ALL\_FIX=set\_tidy\_names(IP\_DATA\_ALL,syntactic=TRUE,quiet=TRUE)

> sapply(IP\_DATA\_ALL\_FIX,typeof)

> library(data.table)

> hist\_country=data.table(Country=unique(IP\_DATA\_ALL\_FIX[is.na(IP\_DATA\_ALL\_FIX['Country'])==0,]$Country))

> setorder(hist\_country,'Country')

> hist\_country\_with\_id=rowid\_to\_column(hist\_country,var='RowIDCountry')

> View(hist\_country\_with\_id)

> IP\_DATA\_Country\_Freq=data.table(with(IP\_DATA\_ALL\_FIX,table(Country)))

> View(IP\_DATA\_Country\_Freq) > sapply(IP\_DATA\_ALL\_FIX[,'Latitude'],min,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Country'],min,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Latitude'],max,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Country'],max,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Latitude'],mean,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Latitude'],median,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Latitude'],range,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Latitude'],quantile,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Latitude'],sd,na.rm=TRUE)

> sapply(IP\_DATA\_ALL\_FIX[,'Longitude'],sd,na.rm=TRUE)

## **Program to retrieve different attributes of data.**

**CODE:**

import pandas as pd

import os

Base='C:/'

sFileName=Base + '/Data Science/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!! ####')

print('Ayush Patel - 5300423035')

## **Data Pattern using R.**

**Example1:-**

**CODE:**

> library(readr)

> library(data.table)

> FileName=('C:/Users/DELL/OneDrive/Desktop/MSc IT-1/DataSciencePractical/VKHCG/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv')

> IP\_DATA\_ALL<-read.csv(FileName)

> IP\_DATA\_ALL<-read\_csv(FileName)

> hist\_country=data.table(Country=unique(IP\_DATA\_ALL$Country))

> pattern\_country=data.table(Country=hist\_country$Country,PatternCountry=hist\_country$Country)

> oldchar=c(letters,LETTERS)

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

> for(r in seq(nrow(pattern\_country))){s=pattern\_country[r,]$PatternCountry;

+ for(c in seq(length(oldchar))){s=chartr(oldchar[c],newchar[c],s)};

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

+ s=chartr(" ","b",s)

+ s=chartr(".","u",s)

+ pattern\_country[r,]$PatternCountry=s;};

> View(pattern\_country)

**Example2:-**

**CODE:-**

> library(readr)

> library(data.table)

> FileName=('C:/Users/DELL/OneDrive/Desktop/MSc IT-1/DataSciencePractical/VKHCG/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv')

> IP\_DATA\_ALL<-read\_csv(FileName)

> hist\_latitude=data.table(Latitude=unique(IP\_DATA\_ALL$Latitude))

> pattern\_latitude=data.table(latitude=hist\_latitude$Latitude,PatternLatitude=as.character(hist\_latitude$Latitude))

> oldchar=c(letters,LETTERS)

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

> for(r in seq(nrow(pattern\_latitude))){s=pattern\_latitude[r,]$PatternLatitude;

+ for(c in seq(length(oldchar))){s=chartr(oldchar[c],newchar[c],s)};

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

+ s=chartr(" ","b",s)

+ s=chartr("+","u",s)

+ s=chartr("-","u",s)

+ s=chartr(".","u",s)

+ pattern\_latitude[r,]$PatternLatitude=s;};

> setorder(pattern\_latitude,latitude)

> View(pattern\_latitude[1:6])

1. **1) Loading IP\_DATA\_ALL using Python**

This data set contains all the IP address allocation in the world. It will help you to locate your customer when interacting with them online.

Create a new python script and save it as Retrieve-IP\_DATA\_ALL.py in directory.

**CODE:**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/1\_UPG\_MSCIT\_PART1'

else:

Base='C:/'

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

sFileName=Base + '/Data Science/IP\_DATA\_ALL.csv'

print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1")

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

sFileDir=Base + '/Data\_Science\_Practical/Practical\_Files'

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!! ############################################')

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

print('Ayush Patel - 53004230035')

## **2) Haversine distance calculating using Vermeulen dataset**

**CODE:**

import os

import pandas as pd

from math import radians,cos,sin,asin,sqrt

print('Ayush Patel - 53004230035')

def haversine(lon1,lat1,lon2,lat2,stype):

'''Calculate the great circle distance between two points on the earth

(specified in decimal degrees)'''

# convert decimal degrees to radians

lon1,lat1,lon2,lat2 = map(radians,[lon1,lat1,lon2,lat2])

# haversine formula

dlon = lon2-lon1

dlat = lat2-lat1

a = sin(dlat/2)\*\*2 + cos(lat1)\*cos(lat2)\*sin(dlon/2)\*\*2

c = 2\*asin(sqrt(a))

if stype == 'km':

r = 6371 # radius of earth in kilometers

else:

r = 3956 # radius of earth in miles

d = round(c\*r,3)

return d

Base = 'C:/'

FileName = Base+'/Data Science/IP\_DATA\_CORE.csv'

IP\_DATA\_CORE = pd.read\_csv(FileName,header=0,low\_memory=False,

usecols=['Country','Place Name','Latitude','Longitude'],encoding='latin-1')

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

if not os.path.exists(FileDir):

os.makedirs(FileDir)

IP\_DATA = IP\_DATA\_CORE.drop\_duplicates(subset=None,keep='first',inplace=False)

IP\_DATA.rename(columns={'Place Name':'Place\_Name'}, inplace=True)

IP\_DATA1 = IP\_DATA

IP\_DATA1.insert(0,'K',1)

IP\_DATA2 = IP\_DATA1

print(IP\_DATA1.shape)

print(IP\_DATA)

IP\_CROSS = pd.merge(right=IP\_DATA1,left=IP\_DATA2,on='K')

print(IP\_CROSS)

IP\_CROSS.drop('K',axis=1,inplace=True)

IP\_CROSS.rename(columns={'Longitude\_x':'Longitude\_from','Longitude\_y':'Longitude\_to'},inplace=True)

IP\_CROSS.rename(columns={'Latitude\_x':'Latitude\_from','Latitude\_y':'Latitude\_to'},inplace=True)

IP\_CROSS.rename(columns={'Place\_Name\_x':'Place\_Name\_from','Place\_Name\_y':'Place\_Name\_to'},inplace=True)

IP\_CROSS.rename(columns={'Country\_x':'Country\_from','Country\_y':'Country\_to'},inplace=True)

print(IP\_CROSS)

IP\_CROSS['DistanceBetweenKilometers'] = IP\_CROSS.apply(lambda row:

haversine(row['Longitude\_from'],

row['Latitude\_from'],

row['Longitude\_to'],

row['Latitude\_to'],'km'),axis=1)

IP\_CROSS['DistanceBetweenMiles'] = IP\_CROSS.apply(lambda row:

haversine(row['Longitude\_from'],

row['Latitude\_from'],

row['Longitude\_to'],

row['Latitude\_to'],'miles'),axis=1)

print(IP\_CROSS.shape)

print('Ayush - 53004230035')

FileName2 = FileDir+'/Retrieve\_IP\_Routing.csv'

IP\_CROSS.to\_csv(FileName2,index=False,encoding='latin-1')

## **Building a Diagram for the Scheduling of Jobs**

Start your Python editor and create a text file named Retrieve-Router-Location.py in directory

**CODE:**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

InputFileName='IP\_DATA\_CORE.csv'

OutputFileName='Retrieve\_Router\_Location.csv'

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/1\_UPG\_MSCIT\_PART1'

else:

Base='C:/'

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

sFileName=Base + '/Data Science/IP\_DATA\_CORE.csv/'

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 + '/Data Science/'

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!! ############################################')

print('Ayush Patel - 53004230035')

## **Picking Content for billboards**

Start your Python editor and create a text file named Retrieve-DE-Billboard-Location,py in directory

**CODE:**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

InputFileName='DE\_Billboard\_Locations.csv'

OutputFileName='Retrieve\_DE\_Billboard\_Locations.csv'

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/1\_UPG\_MSCIT\_PART1'

else:

Base='D:/1\_UPG\_MSCIT\_PART1'

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

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

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

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

Base='C:/'

sFileName=Base + '/Data Science/DE\_Billboard\_Locations.csv'

print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False,

usecols=['Country','PlaceName','Latitude','Longitude'])

IP\_DATA\_ALL.rename(columns={'PlaceName': 'Place\_Name'}, inplace=True)

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

sFileDir=Base + '/Data\_Science\_Practical/Practical\_Files'

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)

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

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

print('Ayush Patel - 53004240035')

## **Understanding your online visitor data**

**CODE:**

#online visitor

import os

import pandas as pd

import gzip as gz

InputFileName = 'IP\_DATA\_ALL.csv'

OutputFileName = 'Retrieve\_Online\_Visitor'

print('Ayush Patel - 53004230035')

Base = 'C:/'

FileName = Base+'/Data Science/IP\_DATA\_ALL.csv'

IP\_DATA\_ALL = pd.read\_csv(FileName,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)

FileDir = Base+'/Data Science/'

if not os.path.exists(FileDir):

os.makedirs(FileDir)

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')

FileName2 = FileDir+'/'+OutputFileName+'.csv'

visitordata.to\_csv(FileName2,index=False)

print('Store All: ',FileName2)

FileName3 = FileDir+'/'+OutputFileName+'\_10.csv'

visitordata10.to\_csv(FileName3,index=False)

print('Store 10: ',FileName3)

for z in ['gzip','bz2','xz']:

if z == 'gzip':

FileName4 = FileName2+'.gz'

else:

FileName4 = FileName2+'.'+z

visitordata.to\_csv(FileName4,index=False,compression=z)

print('Store: ',FileName4)

print('Export JSON')

for Orient in ['split','records','index','columns','values','table']:

FileName2 = FileDir+'/'+OutputFileName+'\_'+Orient+'.json'

visitordata.to\_json(FileName2,orient=Orient,force\_ascii=True)

print('Store All: ',FileName2)

FileName3 = FileDir+'/'+OutputFileName+'10'+Orient+'.json'

visitordata10.to\_json(FileName3,orient=Orient,force\_ascii=True)

print('Store 10: ',FileName3)

FileName4 = FileName2+'.gz'

file\_in = open(FileName2,'rb')

file\_out = gz.open(FileName4,'wb')

file\_out.writelines(file\_in)

file\_in.close()

file\_out.close()

print('Store GZIP All: ',FileName4)

FileName5 = FileDir+'/'+OutputFileName+'\_'+Orient+'\_UnGZip.json'

file\_in = gz.open(FileName4,'rb')

file\_out = open(FileName5,'wb')

file\_out.writelines(file\_in)

file\_in.close()

file\_out.close()

print('Store UNGZIP All: ',FileName5)

1. **XML processing.**

Start Python editor and create a file named Retrieve-Online-Visitor-XML.py in directory

**CODE:**

import os

import pandas as pd

import pandas as pd

import xml.etree.ElementTree as ET

# Defining Function

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'

Base = 'C:/Data Science'

FileName = Base+'/'+InputFileName

IP\_DATA\_ALL = pd.read\_csv(FileName,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)

FileDir = Base+'/'+CompanyOut+'/01-Retrieve/01-EDS/02-Python'

if not os.path.exists(FileDir):

os.makedirs(FileDir)

visitordata = IP\_DATA\_ALL.head(10000)

print('Ayush Patel - 53004230035')

print('Original Subset Data Frame')

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

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

print(visitordata)

print('Export XML')

XML = df2xml(visitordata)

FileName = FileDir+'/'+OutputFileName

file\_out = open(FileName,'wb')

file\_out.write(XML)

file\_out.close()

print('Store XML: ',FileName)

xml\_data = open(FileName,'rb').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)

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)

**I.** **Connecting to other Data Sources**

Program to connect to different data sources.

1. **SQLite**

**CODE:**

import os

import sqlite3 as sq

import pandas as pd

print('Ayush Patel - 53004230035')

Base = 'C:/'

DatabaseName = Base + '/Data Science/Vermeulen.db'

if not os.path.exists(DatabaseName):

os.makedirs(DatabaseName)

conn = sq.connect(DatabaseName)

FileDir = Base+'/Data Science/'

FileName = FileDir + '/Retrieve\_IP\_Data.csv'

IP\_DATA\_ALL\_FIX = pd.read\_csv(FileName,header=0,low\_memory=False)

IP\_DATA\_ALL\_FIX.index.names = ['ROWIDCSV']

Table = 'IP\_DATA\_ALL'

IP\_DATA\_ALL\_FIX.to\_sql(Table,conn,if\_exists='replace')

TestData = pd.read\_sql\_query('SELECT \* FROM IP\_DATA\_ALL',conn)

print('Data Values')

print(TestData)

print()

print('Data Profile')

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

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

1. **Microsoft Excel**

**CODE:**

import os

import pandas as pd

print('Ayush PAtel - 53004230035')

Base = 'C:/'

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

if not os.path.exists(FileDir):

os.makedirs(FileDir)

CurrencyRawData = pd.read\_excel('C:/Data Science/Country\_Currency.xlsx')

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

CurrencyData = CurrencyRawData[Columns]

CurrencyData.rename(columns={'Country or territory':'Country','ISO-4217':'CurrencyCode'},inplace=True)

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

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

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

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

print(CurrencyData)

# **Practical 4**

**AIM: Assessing Data**

## **Program error management on the given data using pandas’ package**

### **Drop the Columns Where All Elements are Missing Values**

**CODE:**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/1\_UPG\_MSCIT\_PART1'

else:

Base='D:/1\_UPG\_MSCIT\_PART1'

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

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

print('Ayush Patel - 53004230035')

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

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

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

sInputFileName='Good-or-Bad.csv'

sOutputFileName='Good-or-Bad-01.csv'

Company='01-Vermeulen'

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

Base='C:/'

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

sFileDir=Base + '/Data Science/Good-or-Bad.csv'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

### Import Warehouse

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

sFileName=Base + '/Data Science/Good-or-Bad.csv'

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 + 'Good-or-Bad.csv'

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('Ayush Patel - 53004230035')

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

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

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

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

sFileName=sFileDir + 'Good-or-Bad-01.csv'

TestData.to\_csv(sFileName, index = False)

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

print('Ayush Patel - 53004230035')

### **Drop the Columns Where any of the Elements is Missing Values**

Code

**CODE:**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

Base='C:/'

sInputFileName='Good-or-Bad.csv'

sOutputFileName='Good-or-Bad-02.csv'

Company='01-Vermeulen'

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/1\_UPG\_MSCIT\_PART1'

else:

Base='C:/'

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

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

print('Ayush PAtel - 53004230035')

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

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

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

sFileDir=Base + '/Data Science/Good-or-Bad.csv'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

### Import Warehouse

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

sFileName=Base + '/Data Science/Good-or-Bad.csv'

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 + 'Good-or-Bad.csv'

RawData.to\_csv(sFileName, index = False)

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

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

print('## Test Data Values')

print(TestData)

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

print('## Data Profile')

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

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

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

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

sFileName=sFileDir + 'Good-or-Bad-02.csv'

TestData.to\_csv(sFileName, index = False)

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

print('Ayush PAtel - 53004230035')

### **Keep only the rows that contain a maximum of two Missing Values**

Code

##################### Assess-Good-Bad-03.py ################

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

sInputFileName='Good-or-Bad.csv'

sOutputFileName='Good-or-Bad-03.csv'

Base='C:/'

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

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

print('Ayush PAtel -53004230035')

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

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

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

sFileDir=Base + '/Data Science/Good-or-Bad.csv'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

### Import Warehouse

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

sFileName=Base + '/Data Science/Good-or-Bad.csv'

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 + 'Good-or-Bad.csv'

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])

sFileName=sFileDir + 'Good-or-Bad-03.csv'

TestData.to\_csv(sFileName, index = False)

print('### Done!! ')

print('Ayush PAtel -53004230035')

### **Fill all Missing Values with the Mean, Median, Mode, Minimum and Maximum of the Particular Numeric Column**

Code

import sys

import os

import pandas as pd

Base='D:/1\_UPG\_MSCIT\_PART1'

print('AIM: Fill all the missing values with Mean Median Mode Minimum Maximum of numeric column\n ')

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

print(' Ayush Patel - 53004230035 ')

sInputFileName='Good-or-Bad.csv'

sOutputFileNameA='Good-or-Bad-04-A.csv' #MEAN

sOutputFileNameB='Good-or-Bad-04-B.csv' #MEDIAN

sOutputFileNameC='Good-or-Bad-04-C.csv' #MODE

sOutputFileNameD='Good-or-Bad-04-D.csv' #MIN

sOutputFileNameE='Good-or-Bad-04-E.csv' #MAX

Base='C:/'

sFileDir=Base + '/Data Science/Good-or-Bad.csv'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileName=Base + '/Data Science/Good-or-Bad.csv'

print('Loading :',sFileName)

RawData=pd.read\_csv(sFileName,header=0)

print(' ')

print('Raw Data Values')

print(RawData)

print(' ')

print('Data Profile')

print(' ')

print('R-ows :',RawData.shape[0])

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

sFileName=sFileDir + '/' + sInputFileName

RawData.to\_csv(sFileName, index = False)

TestData=RawData.fillna(RawData.mean())

print(' ')

print('## Test Data Values')

print(TestData)

print(' ')

print('## Data Profile')

print(' ')

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

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

sFileName=sFileDir + '/' + sOutputFileNameA

TestData.to\_csv(sFileName, index = False)

TestData=RawData.fillna(RawData.median())

print(' ')

print('## Test Data Values')

print(TestData)

print(' ')

print('## Data Profile')

print(' ')

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

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

sFileName=sFileDir + '/' + sOutputFileNameB

TestData.to\_csv(sFileName, index = False)

TestData=RawData.fillna(RawData.mode())

print(' ')

print('## Test Data Values')

print(TestData)

print(' ')

print('## Data Profile')

print(' ')

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

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

print(' ')

sFileName=sFileDir + '/' + sOutputFileNameC

TestData.to\_csv(sFileName, index = False)

TestData=RawData.fillna(RawData.min())

print(' ')

print('## Test Data Values')

print(TestData)

print(' ')

print('## Data Profile')

print(' ')

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

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

print(' ')

sFileName=sFileDir + '/' + sOutputFileNameD

TestData.to\_csv(sFileName, index = False)

TestData=RawData.fillna(RawData.max())

print(' ')

print('## Test Data Values')

print(TestData)

print(' ')

print('## Data Profile')

print(' ')

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

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

print(' ')

sFileName=sFileDir + '/' + sOutputFileNameE

TestData.to\_csv(sFileName, index = False)

print('Done!!!!!')

print(' ')

# **Practical 5**

**AIM: Processing Data**

## **Build the time hub, links and satellites.**

Code

import os

import pandas as pd

import sqlite3 as sq

from datetime import datetime,timedelta

from pytz import timezone,all\_timezones

from pandas.io import sql

import uuid

pd.options.mode.chained\_assignment = None

Base = 'C:/Data Science'

InputFileName = 'VehicleData.csv'

DataBaseDir = Base+'/''/03-Process/SQLite'

if not os.path.exists(DataBaseDir):

os.makedirs(DataBaseDir)

DataBaseName = DataBaseDir+'/Hillman.db'

conn1 = sq.connect(DataBaseName)

DataVaultDir = Base+'/88-DV'

if not os.path.exists(DataBaseDir):

os.makedirs(DataBaseDir)

DataBaseName = DataVaultDir+'/datavault.db'

conn2 = sq.connect(DataBaseName)

base = datetime(2018,1,1,0,0,0)

numUnits = 1000

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'))

DateTime = now\_utc.strftime('%Y-%m-%d %H:%M:%S')

DateTimeKey = DateTime.replace(' ','-').replace(':','-')

t+=1

IDNumber = str(uuid.uuid4())

TimeLine = {'ZoneBaseKey':['UTC'],

'IDNumber':[IDNumber],

'nDateTimeValue':[now\_utc],

'DateTimeValue':[DateTime],

'DateTimeKey':[DateTimeKey]}

if t==1:

TimeFrame = pd.DataFrame.from\_dict(TimeLine)

else:

TimeRow = pd.DataFrame.from\_dict(TimeLine)

TimeFrame = pd.concat([TimeFrame,TimeRow],ignore\_index=True)

TimeHub = TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]

TimeHubIndex = TimeHub.set\_index(['IDNumber'],inplace=False)

TimeFrame.set\_index(['IDNumber'],inplace=True)

Table = 'Process-Time'

TimeHubIndex.to\_sql(Table,conn1,if\_exists='replace')

Table = 'Hub-Time'

TimeHubIndex.to\_sql(Table,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'))

DateTime=now\_utc.strftime("%Y-%m-%d %H:%M:%S")

now\_zone = now\_utc.astimezone(timezone(zone))

ZoneDateTime=now\_zone.strftime("%Y-%m-%d %H:%M:%S")

print(ZoneDateTime)

t+=1

z+=1

IDZoneNumber=str(uuid.uuid4())

TimeZoneLine={'ZoneBaseKey': ['UTC'],

'IDZoneNumber': [IDZoneNumber],

'DateTimeKey': [DateTimeKey],

'UTCDateTimeValue': [DateTime],

'Zone': [zone],

'DateTimeValue': [ZoneDateTime]}

if t==1:

TimeZoneFrame = pd.DataFrame.from\_dict(TimeZoneLine)

else:

TimeZoneRow = pd.DataFrame.from\_dict(TimeZoneLine)

TimeZoneFrame = pd.concat([TimeZoneFrame,TimeZoneRow],ignore\_index=True)

TimeZoneFrameIndex = TimeZoneFrame.set\_index(['IDZoneNumber'],inplace=False)

Zone = zone.replace('/','-').replace(' ','-')

Table = 'Process-Time-'+Zone

TimeZoneFrameIndex.to\_sql(Table,conn1,if\_exists='replace')

Table = 'Satellite-Time-'+Zone

TimeZoneFrameIndex.to\_sql(Table,conn2,if\_exists='replace')

print('VACUUM Databases')

SQL = ' VACUUM;'

sql.execute(SQL,conn1)

sql.execute(SQL,conn2)

## **Golden Nominal.**

Code

import os

import sqlite3 as sq

import pandas as pd

from datetime import datetime, timedelta

from pytz import timezone, all\_timezones

from random import randint

print('Ayush - 53004230035')

Base = 'C:/'

Inputfilename = '/Assess\_People.csv'

DatabaseDir = Base+'/Data Science'

if not os.path.exists(DatabaseDir):

os.makedirs(DatabaseDir)

Databasename = DatabaseDir+'/clark.db'

conn = sq.connect(Databasename)

Filename = Base+'/Data Science/Assess\_People.csv'

print('Loading: ',Filename)

RawData = pd.read\_csv(Filename,header=0,low\_memory=False,encoding='latin-1')

print(RawData)

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)

Data = RawData.copy()

Data.drop('BirthDateUTC',axis=1,inplace=True)

Data.drop('BirthDateISO',axis=1,inplace=True)

Table = 'Process\_Person'

Data.to\_sql(Table,conn,if\_exists='replace')

FileDir = Base+'/Data Science'

if not os.path.exists(FileDir):

os.makedirs(FileDir)

OutputFileName = Table+'.csv'

Filename = FileDir+'/'+OutputFileName

RawData.to\_csv(Filename,index=False)

**Practical 6**

**AIM: Transforming Data**

## **Write python program to demonstrate Simple Linear Regression.**

Code

import os

import sqlite3 as sq

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets,linear\_model

from sklearn.metrics import mean\_squared\_error,r2\_score

# Load Diabetes Data

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='red',linewidth=3)

plt.xticks(())

plt.yticks(())

plt.axis('tight')

plt.title('Diabetes')

plt.xlabel('BMI')

plt.ylabel('Age')

plt.show()

print("Ayush PAtel - 53004230035")

# **Practical 7**

**AIM: Organizing Data**

**A - Write Python program to perform the horizontal style subset of slice of the data warehouse data.**

Code

import sys

import os

import pandas as pd

import sqlite3 as sq

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '1\_UPG\_MSCIT\_PART1'

else:

Base='C:/'

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

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

print('Ayush Patel - 53004230035')

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

sDataWarehouseDir=Base + '/Data Science'

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 PersonID,\

Height,\

Weight,\

bmi,\

Indicator\

FROM [Dim-BMI]\

WHERE \

Height > 1.5 \

and Indicator = 1\

ORDER BY \

Height,\

Weight;"

PersonFrame1=pd.read\_sql\_query(sSQL, conn1)

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

DimPerson=PersonFrame1

DimPersonIndex=DimPerson.set\_index(['PersonID'],inplace=False)

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

sTable = 'Dim-BMI-Horizontal'

print('\n#################################')

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

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

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

sTable = 'Dim-BMI-Horizontal'

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

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

sSQL="SELECT \* FROM [Dim-BMI];"

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

## **Write Python program to perform the vertical style subset of slice of the data warehouse data.**

Code

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

import sqlite3 as sq

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/1\_UPG\_MSCIT\_PART1'

else:

Base='C:/'

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

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

print('Ayush Patel - 53004230035')

sDataWarehouseDir=Base + '/Data Science'

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

## **Write Python program to perform the island style subset of slice of the data warehouse data.**

Code

import sys

import os

import pandas as pd

import sqlite3 as sq

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/1\_UPG\_MSCIT\_PART1'

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

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

Company='01-Vermeulen'

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

sDataWarehouseDir=Base + '/Data Science'

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

## **Write Python program to perform the secure vault style subset of slice of the data warehouse data and attach the result to the person who performs the query.**

Code

import sys

import os

import pandas as pd

import sqlite3 as sq

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/1\_UPG\_MSCIT\_PART1'

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

Company='01-Vermeulen'

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

sDataWarehouseDir=Base + '/Data Science'

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

## **Write Python program to demonstrate 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

print('Ayush Patel - 53004230035')

Base = 'C:/Data Science'

Company='01-Vermeulen'

InputFileName='Online-Retail-Billboard.xlsx'

EDSAssessDir='02-Assess/01-EDS'

InputAssessDir=EDSAssessDir + '/02-Python'

FileAssessDir=Base + '/' + Company + '/' + InputAssessDir

if not os.path.exists(FileAssessDir):

os.makedirs(FileAssessDir)

FileName=Base+'/' + InputFileName

df = pd.read\_excel(FileName)

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) ]

Product1='ALARM CLOCK BAKELIKE GREEN'

print(Product1)

print(basket[Product1].sum())

Product2='ALARM CLOCK BAKELIKE RED'

print(Product2)

print(basket[Product2].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)])

## **F - Write Python program to create network routing diagram in organizing superstep.**

Code

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

import sys

import os

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

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

pd.options.mode.chained\_assignment = None

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '1\_UPG\_MSCIT\_PART1'

else:

Base='C:/'

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

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

print('UPGCM')

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

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

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

sInputFileName='Assess-Network-Routing-Company.csv'

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

sOutputFileName1='Organise-Network-Routing-Company.gml'

sOutputFileName2='Organise-Network-Routing-Company.png'

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

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

### Import Country Data

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

sFileName=Base + '/Data Science/Assess-Network-Routing-Company.csv'

print('Ayush Patel - 53004230035')

print('Loading :',sFileName)

CompanyData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1")

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

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 + '/Data\_Science\_Practical/Practical\_Files/' + sOutputFileName1

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

print('Storing :',sFileName)

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

nx.write\_gml(G, sFileName)

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

sFileName=Base + '/Data\_Science\_Practical/Practical\_Files/' + 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('################################')

# **Practical 8**

**AIM: Generating Data**

**A - Write a python program to perform data visualization to create following graphs using profit data.**

1) Pie Graph

**CODE:**

import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush PAtel - 53004230035')

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

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

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

GBase = Base+'/Data Science/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os\_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'ConversionPercent','URL', 'Stats URL'],dtype='object')

os\_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os\_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os\_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

explode = (0, 0, 0, 0, 0, 0.1)

labels=os\_new['Warehouse Location']

colors\_mine = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral',

'lightcyan','lightblue']

os\_new.plot(figsize=(10, 10),kind="pie", y="Profit 2017",autopct='%.2f%%', \

shadow=True, explode=explode, legend = False, colors = colors\_mine,\

labels=labels, fontsize=20)

sPicNameOut1=GBase+'pie\_explode.png'

plt.savefig(sPicNameOut1,dpi=600)

2) Double Pie Graph

**CODE:**

import matplotlib.pyplot as plt

Base = 'C:/Data Science/'

data = [['London',29.2,17.4],

['Glasgow',18.8,11.3],

['Cape Town',15.3,9.0],

['Houston',22.0,7.8],

['Perth',18.0,23.7],

['San Francisco',11.4,33.3]]

labels = ['London', 'Glasgow', 'Cape Town', 'Houston', 'Perth', 'San Francisco']

profit\_2016 = [29.2, 18.8, 15.3, 22.0, 18.0, 11.4]

profit\_2017 = [17.4, 11.3, 9.0, 7.8, 23.7, 33.3]

colors\_mine = ['yellow','red','blue','green','orange','pink']

# Create a figure with two subplots (one row, two columns)

fig, axs = plt.subplots(1, 2, figsize=(14, 7))

# Plot the first pie chart for Profit 2016

axs[0].pie(profit\_2016, labels=labels, autopct='%.2f%%', shadow=False, startangle=90, colors=colors\_mine)

axs[0].set\_title('Profit 2016')

# Plot the second pie chart for Profit 2017

axs[1].pie(profit\_2017, labels=labels, autopct='%.2f%%', shadow=False, startangle=90, colors=colors\_mine)

axs[1].set\_title('Profit 2017')

plt.tight\_layout()

Picnameout2 = Base + 'doublepie.png'

plt.savefig(Picnameout2, dpi=500)

plt.show()

print('Ayush - 35')

3) Line Graph

**CODE:**

import os

import pandas as pd

import matplotlib as ml

import matplotlib.pyplot as plt

Base = 'C:/Data Science/'

ml.style.use('ggplot')

data = [['London',29.2,17.4],

['Glasgow',18.8,11.3],

['Cape Town',15.3,9.0],

['Houston',22.0,7.8],

['Perth',18.0,23.7],

['San Francisco',11.4,33.3]]

new = pd.DataFrame(data)

new.rename(columns={0:'WareHouse Location'},inplace=True)

new.rename(columns={1:'Profit 2016'},inplace=True)

new.rename(columns={2:'Profit 2017'},inplace=True)

# Pie Graph

explode = (0,0,0,0,0,0.1)

labels = new['WareHouse Location']

colors\_mine = ['yellow','red','blue','green','black','pink']

new.iloc[:5].plot(figsize=(10,10),kind='line',x='WareHouse Location',y=['Profit 2016','Profit 2017'])

Picnameout3 = Base+'Line.png'

plt.savefig(Picnameout3,dpi=600)

print('Ayush - 35')

4) Bar Graph

**CODE:**

import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

GBase = Base+'/Data Science/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os\_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'ConversionPercent','URL', 'Stats URL'],dtype='object')

os\_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os\_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os\_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os\_new.iloc[:5].plot(figsize=(10, 10),kind='bar',x='Warehouse Location',\

y=['Profit 2016','Profit 2017']);

sPicNameOut4=GBase+'bar.png'

plt.savefig(sPicNameOut4,dpi=600)

5) Horizontal Bar Graph

**CODE:**

import sys

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

GBase = Base+'/Data Science/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os\_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'ConversionPercent','URL', 'Stats URL'],dtype='object')

os\_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os\_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os\_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os\_new.iloc[:5].plot(figsize=(10, 10),kind='barh',x='Warehouse Location',\

y=['Profit 2016','Profit 2017']);

sPicNameOut5=GBase+'hbar.png'

plt.savefig(sPicNameOut5,dpi=600)

6) Area Graph

**CODE:**

import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

GBase = Base+'/Data Science/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os\_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'ConversionPercent','URL', 'Stats URL'],dtype='object')

os\_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os\_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os\_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os\_new.iloc[:5].plot(figsize=(10, 10),kind='area',x='Warehouse Location',\

y=['Profit 2016','Profit 2017'],stacked=False);

sPicNameOut6=GBase+'area.png'

plt.savefig(sPicNameOut6,dpi=600)

7) Scatter Graph

**CODE:**

import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

GBase = Base+'/Data Sciences/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os\_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'ConversionPercent','URL', 'Stats URL'],dtype='object')

os\_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os\_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os\_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os\_new.iloc[:5].plot(figsize=(10, 10),kind='scatter',x='Profit 2016',\

y='Profit 2017',color='DarkBlue',marker='D');

sPicNameOut7=GBase+'scatter.png'

plt.savefig(sPicNameOut7,dpi=600)

8) HexBin Graph

**CODE:**

import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

GBase = Base+'/Data Science/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]]

os\_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'ConversionPercent','URL', 'Stats URL'],dtype='object')

os\_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os\_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os\_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

os\_new.iloc[:5].plot(figsize=(13, 10),kind='hexbin',x='Profit 2016',\

y='Profit 2017', gridsize=25);

sPicNameOut8=GBase+'hexbin.png'

plt.savefig(sPicNameOut8,dpi=600)

## **b - Write a python program to perform data visualization to create following advanced graphs/plots using profit data.**

1) Kernel Density Estimation (KDE) Graph

**Code**

import sys

import os

import pandas as pd

import matplotlib as ml

import numpy as np

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patyel - 53004230035')

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

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

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

ml.style.use('ggplot')

fig1=plt.figure(figsize=(10, 10))

ser = pd.Series(np.random.randn(1000))

ser.plot(figsize=(10, 10),kind='kde')

sPicNameOut1=Base+'//Data Science/kde.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight\_layout()

plt.show()

2) Scatter Matrix Graph

**CODE:**

import sys

import os

import pandas as pd

import matplotlib as ml

import numpy as np

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel 53004230035')

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

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

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

ml.style.use('ggplot')

fig2=plt.figure(figsize=(10, 10))

from pandas.plotting import scatter\_matrix

df = pd.DataFrame(np.random.randn(1000, 5), columns=['Y2014','Y2015',

'Y2016', 'Y2017', 'Y2018'])

scatter\_matrix(df, alpha=0.2, figsize=(10, 10), diagonal='kde')

sPicNameOut2=Base+'/Data Science/scatter\_matrix.png'

plt.savefig(sPicNameOut2,dpi=600)

plt.tight\_layout()

plt.show()

3) Andrew’s Curves

**CODE:**

import sys

import os

import pandas as pd

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

sDataFile=Base+'/Data Science/irisdata.csv'

data = pd.read\_csv(sDataFile)

from pandas.plotting import andrews\_curves

plt.figure(figsize=(10, 10))

andrews\_curves(data, 'Name')

sPicNameOut1=Base+'/Data Science/andrews\_curves.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight\_layout()

plt.show()

4) Parallel Coordinates

**CODE:**

import sys

import os

import pandas as pd

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

sDataFile=Base+'/Data Science/irisdata.csv'

data = pd.read\_csv(sDataFile)

from pandas.plotting import parallel\_coordinates

plt.figure(figsize=(10, 10))

parallel\_coordinates(data, 'Name')

sPicNameOut2=Base+'/Data Science/parallel\_coordinates.png'

plt.savefig(sPicNameOut2,dpi=600)

plt.tight\_layout()

plt.show()

5) RADVIZ Method

**CODE:**

import sys

import os

import pandas as pd

from matplotlib import pyplot as plt

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

sDataFile=Base+'/Data Science/irisdata.csv'

data = pd.read\_csv(sDataFile)

from pandas.plotting import radviz

plt.figure(figsize=(10, 10))

radviz(data, 'Name')

sPicNameOut3=Base+'/Data Science/radviz.png'

plt.savefig(sPicNameOut3,dpi=600)

plt.tight\_layout()

plt.show()

6) Lag Plot

**CODE:**

import sys

import os

import pandas as pd

from matplotlib import style

from matplotlib import pyplot as plt

import numpy as np

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

style.use('ggplot')

from pandas.plotting import lag\_plot

plt.figure(figsize=(10, 10))

data = pd.Series(0.1 \* np.random.rand(1000) + \

0.9 \* np.sin(np.linspace(-99 \* np.pi, 99 \* np.pi, num=1000)))

lag\_plot(data)

sPicNameOut1=Base+'/Data Science/lag\_plot.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight\_layout()

plt.show()

7) AutoCorrelation Plot

**CODE:**

import os

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

from matplotlib import style

# Lag Plot

Base = 'C:/Data Science'

style.use('ggplot')

from pandas.plotting import autocorrelation\_plot

plt.figure(figsize=(10,10))

data = pd.Series(0.7\*np.random.rand(1000) + \

0.3\*np.sin(np.linspace(-9\*np.pi,9\*np.pi,num=1000)))

autocorrelation\_plot(data)

Picnameout7 = Base+'/autocorrelation\_plot.png'

plt.savefig(Picnameout7,dpi=600)

plt.tight\_layout()

plt.show()

print('Ayush - 35')

8) Bootstrap Plot

**CODE:**

import sys

import os

import pandas as pd

from matplotlib import style

from matplotlib import pyplot as plt

import numpy as np

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

if sys.platform == 'linux' or sys.platform == ' darwin':

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

else:

Base='C:/'

print('Ayush Patel - 53004230035')

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

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

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

style.use('ggplot')

from pandas.plotting import bootstrap\_plot

data = pd.Series(np.random.rand(1000))

plt.figure(figsize=(10, 10))

bootstrap\_plot(data, size=50, samples=500, color='grey')

sPicNameOut3=Base+'/Data Science/bootstrap\_plot.png'

plt.savefig(sPicNameOut3,dpi=600)

plt.tight\_layout()

plt.show()

9) Contour Graph

**CODE:**

import os

import numpy as np

import matplotlib

import matplotlib.cm as cm

import matplotlib.pyplot as plt

def bivariate\_normal(X,Y,sigmax=1.0,sigmay=1.0,mux=0.0,muy=0.0,sigmaxy=0.0):

Xmu = X-mux

Ymu = Y-muy

rho = sigmaxy/(sigmax\*sigmay)

z = Xmu\*\*2/sigmax\*\*2 + Ymu\*\*2/sigmay\*\*2 - 2\*rho\*Xmu\*Ymu/(sigmax\*sigmay)

denom = 2\*np.pi\*sigmax\*sigmay\*np.sqrt(1-rho\*\*2)

return np.exp(-z/(2\*(1-rho\*\*2)))/denom

delta = 0.025

x = np.arange(-3.0,3.0,delta)

y = np.arange(-2.0,2.0,delta)

X,Y = np.meshgrid(x,y)

Z1 = bivariate\_normal(X,Y,1.0,1.0,0.0,0.0)

Z2 = bivariate\_normal(X,Y,1.5,0.5,1,1)

Z = 10.0\*(Z2-Z1)

plt.figure(figsize=(10,10))

CS = plt.contour(X, Y, Z)

plt.clabel(CS, inline=1, fontsize=10)

plt.title('Simply default with labels')

plt.tight\_layout()

plt.show()

print("Ayush Patel - 53004230035")

10) 3D Graph

**CODE:**

import os

import numpy as np

from matplotlib import pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

from sklearn import decomposition

from sklearn import datasets

# 3D Graphs

np.random.seed(5)

centers = [[1,1],[-1,-1],[1,-1]]

iris = datasets.load\_iris()

X = iris.data

y = iris.target

fig = plt.figure(1,figsize=(16,12))

plt.clf()

ax = Axes3D(fig,rect=[0,0,0.95,1],elev=48,azim=134)

plt.cla()

pca = decomposition.PCA(n\_components=3)

pca.fit(X)

X = pca.transform(X)

for name,label in [('Setosa',0),('Versicolour',1),('Virginica',2)]:

ax.text3D(X[y == label,0].mean(),

X[y == label,1].mean()+1.5,

X[y == label,2].mean(),name,

horizontalalignment='center',

bbox = dict(alpha=0.5, edgecolor='w', facecolor='w'))

y = np.choose(y,[1,2,0]).astype(np.float)

ax.scatter(X[:,0],X[:,1],X[:,2],c=y,edgecolor='k',marker='p',s=300)

ax.w\_xaxis.set\_ticklabels([])

ax.w\_yaxis.set\_ticklabels([])

ax.w\_zaxis.set\_ticklabels([])

Picnameout10 = Base+'/3Dplot.png'

plt.savefig(Picnameout10,dpi=600)

plt.show()

# **Practical 9**

**AIM: Data Visualization with Power BI**

**Case Study: Sales Data**

**Step 1: Connect to an Excel workbook**

1. Launch Power BI Desktop
2. From the Home ribbon, select **Get Data**. Excel is one of the **Most Common** data Collections, so you can select it directly from the **Get Data** menu.
3. If you select the Get Data button directly, you can also select File> Excel and select Connect.
4. In the Open File dialog box, select the Products.xlsx file.

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

1. In Query Editor, select the ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns

*(use Ctrl+ Click to select more than one columns , or Shift+Click to select columns that are beside each other)*

1. Select Remove Columns -> Remove Other Columns from the ribbon, or right-click on a columns header and click Remove Other Columns.
2. Change the data type of the UnitsInStock column

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

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

**Task 2: Import order data from an OData feed**

You import data into Power BI Desktop from the sample Northwind OData feed at the following

URL, which you can copy (and then paste) in the steps below:

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

**Step 1: Connect to an OData feed**

1. From the **Home** ribbon tab in Query Editor, select **Get Data**.

2. Browse to the **OData Feed** data source.

3. In the **OData Feed** dialog box, paste the **URL** for the Northwind OData feed.

4. Select **OK**.

.

**Step 2: Expand the Order\_Details table**

Expand the **Order\_Details** table that is related to the **Orders** table, to combine the **ProductID,**

**UnitPrice**, and **Quantity** columns from **Order\_Details** into the **Orders** table.

The **Expand** operation combines columns from a related table into a subject table. When the query runs, rows from the related table **(Order\_Details)** are combined into rows from the subject table

**(Orders).**

After you expand the **Order\_Details** table, three new columns and additional rows are added to the

**Orders** table, one for each row in the nested or related table.

1. In the **Query View,** scroll to the **Order\_Details** column.

2. In the **Order\_Details** column, select the expand icon ().

3. In the **Expand** drop-down: a. Select **(Select All Columns)** to clear all columns.

Select **ProductID, UnitPrice,** and **Quantity.**

click **OK**.

**Step 3: Remove other columns to only display columns of interest**

In this step you remove all columns except **OrderDate, ShipCity, ShipCountry,**

**Order\_Details.ProductID, Order\_Details.UnitPrice,** and **Order\_Details.Quantity** columns. In the previous task, you used **Remove Other Columns**. For this task, you remove selected columns.

In the **Query View**, select all columns by completing a.

a. Click the first column **(OrderID)**.

b. Shift+Click the last column **(Shipper**).

c. Now that all columns are selected, use Ctrl+Click to unselect the following columns:

OrderDate, ShipCity, ShipCountry, Order\_Details.ProductID, Order\_Details.UnitPrice, and

Order\_Details.Quantity.

Now that only the columns we want to remove are selected, right-click on any selected column

header and click Remove Columns.

**Step 4: Calculate the line total for each Order\_Details row**

Power BI Desktop lets you to create calculations based on the columns you are importing, so you can

enrich the data that you connect to. In this step, you create a **Custom Column** to calculate the line

total for each **Order\_Details** row.

Calculate the line total for each **Order\_Details** row:

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

2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter

**[Order\_Details.UnitPrice] \* [Order\_Details.Quantity].**

3. In the New column name textbox, enter LineTotal.

**Step 5: Set the datatype of the LineTotal field**

1. Right click the **LineTotal** column.

2. Select **Change Type** and choose **Decimal Number**.

**Step 6: Rename and reorder columns in the query**

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

2. Remove

2. Remove the Order\_Details. prefix from the **Order\_Details.ProductID, Order\_Details.UnitPrice**

and **Order\_Details.Quantity** columns, by double-clicking on each column header, and then deleting

that text from the column name.

**Task 3: Combine the Products and Total Sales queries**

1.First, we need to load the model that we created in Query Editor into Power BI Deshtop. From the **Home** Ribbon of Query Editor, select **Close & Load**.

2. Power BI Desktop loads the data from the two queries

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

4. Select the New… button

5. When we attempt to create the relationship, we see that one already exists! As shown in the

Create Relationship dialog (by the shaded columns), the ProductsID fields in each query

already have an established relationship

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

**Task 4: Build visuals using your data**

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

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

Step 3: Next, drag ShipCountry to a spare on the canvas in the top right. Because you selected a geographic field , a map was created automatically . Now drag LineTotal to the Values field; the circles on the map for each country are now relative in size to the LineTotal for orders shipped to that country